```c
/*  Name: Rodrigo Ignacio Rojas Garcia
    Lab #7
*/

// Library Declaration Section
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <math.h>

// Define Declaration Section
#define MAXLENGTH 256
#define ACC_THRESHOLD_X 0.0009
#define ACC_THRESHOLD_Y 0.0009
#define ACC_THRESHOLD_Z 0.0009
#define GYRO_THRESHOLD_PITCH 0.03
#define GYRO_THRESHOLD_ROLL 0.03
#define GYRO_THRESHOLD_YAW 0.03
#define SMOOTH_WINDOW_SIZE 25
#define VARIANCE_WINDOW_SIZE 11
#define SAMPLE_TIME 0.05
#define GRAVITY 9.81

// READ IN TEXT FILE AND EXTRACT DATA
void read_text_file(char *file_name, int *file_size, double **time, double **accX,
double **accY,
                    double **accZ, double **pitch, double **roll, double **yaw)
{
    // VARIABLE DECLARATION SECTION
    FILE *file;
    int i = 0;
    double d1, d2, d3, d4, d5, d6, d7;
    char c;
    char line[MAXLENGTH];
    *file_size = 0;

    // OBTAINS FILE LENGTH AND REWINDS IT TO BEGINNING
    file = fopen(file_name, "r");
    if (file == NULL)
    {
        printf("Error, could not read in initial contour text file\n");
        exit(1);
    }
    while((c = fgetc(file)) != EOF)
    {
        if (c == '\n')
        {
            *file_size += 1;
        }
    }
    rewind(file);

    // SKIPS FIRST LINE OF FILE
    fgets(line, sizeof(line), file);

    // ALLOCATES MEMORY
    *time = calloc(*file_size, sizeof(double *));
    *accX = calloc(*file_size, sizeof(double *));
    *accY = calloc(*file_size, sizeof(double *));
    *accZ = calloc(*file_size, sizeof(double *));
    *pitch = calloc(*file_size, sizeof(double *));
    *roll = calloc(*file_size, sizeof(double *));
    *yaw = calloc(*file_size, sizeof(double *));

    // EXTRACTS TDATA FROM TEXT FILE
    while((fscanf(file, "%lf %lf %lf %lf %lf %lf %lf\n", &d1, &d2, &d3, &d4, &d5,
&d6, &d7)) != EOF)
    {
        (*time)[i] = d1;
        (*accX)[i] = d2;
        (*accY)[i] = d3;
        (*accZ)[i] = d4;
        (*pitch)[i] = d5;
        (*roll)[i] = d6;
        (*yaw)[i] = d7;
        i++;
    }
    fclose(file);

    // CREATE CSV FILE
    file = fopen("initial-data.csv", "w");
    fprintf(file, "Time[s],Acceletarion-X[m/s],Acceletarion-Y[m/s],Acceleration-Z[m/
s],Pitch,Roll,Yaw\n");
    for (i = 0; i < *file_size; i++)
    {
        fprintf(file, "%lf,%lf,%lf,%lf,%lf,%lf,%lf\n", (*time)[i], (*accX)[i], (*accY)
[i], (*accZ)[i], (*pitch)[i], (*roll)[i], (*yaw)[i]);
    }
    fclose(file);
}

// SMOOTH DATA POINTS
void smooth_data(int arr_length, double **data, double **smoothed_data)
{
    // VARIABLE DECLARATION SECTION
    int i, j;
    double sum;

    // ALLOCATE MEMORY
    *smoothed_data = calloc(arr_length, sizeof(double *));

    for (i = 0; i < SMOOTH_WINDOW_SIZE; i++)
    {
        (*smoothed_data)[i] = (*data)[i];
    }

    for (i = SMOOTH_WINDOW_SIZE - 1; i < arr_length; i++)
    {
        sum = 0;
        for (j = 1; j < SMOOTH_WINDOW_SIZE; j++)
        {
            sum += (*data)[i - j];
        }
        (*smoothed_data)[i] = (sum + (*data)[i]) / SMOOTH_WINDOW_SIZE;
    }
}

// CALCULATE VARIANCE
double calc_variance(double **data, int arr_length, int index)
{
    // VARIABLE DECLARATION SECTION
    int i;
    int window = 0;
    double mean = 0;
```

```c
    double variance = 0;

    // HANDLES ERROR WHEN WINDOW SIZE IS TOO BIG
    if ((index + VARIANCE_WINDOW_SIZE) < arr_length)
    {
        window = VARIANCE_WINDOW_SIZE;
    }
    else
    {
        window = abs(arr_length - index);
    }

    // CALCULATE THE SUM OF DATA SET
    for (i = index; i < (index + window); i++)
    {
        variance += (*data)[i];
    }

    // USED SUM CALCULATED TO OBTAIN MEAN OF DATA SET
    mean = variance / window;
    variance = 0;

    // SUBTRACT MEAN FOR THE DATA SET AND SUM RESULTS
    for (i = index; i < (index + window); i++)
    {
        variance += pow(((*data)[i] - mean), 2);
    }

    // CALCULATE VARIANCE OF EACH DATA SET
    variance = variance / (window - 1);

    return variance;
}

// DETERMINES IF OBJECT HAS MOVED AND STORES RESULTS ON TEXT AND CSV FILES
void move_or_still(double **accX, double **accY, double **accZ, double **pitch, double
**roll, double **yaw, int arr_length)
{
    // VARIABLE DECLARATION SECTION
    FILE *text_file, *csv_file;
    int i, j, k;
    int start_movement = 0;
    int end_movement = 0;
    int moving = 0;
    double start_time = 0;
    double end_time = 0;
    double acc_variance[3];
    double gyro_variance[3];
    double gyro_distance[3];
    double velocity[3];
    double distance_traveled[3];
    double previous_velocity[3];
    double sum[6];
    char file_name[MAXLENGTH];
    k = 1;

    sprintf(file_name, "movement-%d.txt", VARIANCE_WINDOW_SIZE);
    text_file = fopen(file_name, "w");
    sprintf(file_name, "movement-%d.csv", VARIANCE_WINDOW_SIZE);
    csv_file = fopen(file_name, "w");
    fprintf(csv_file, "Start Index,Stop Index,Start Time,End Time,X [m],Y [m],Z
[m],Pitch [radians],Roll [radians],Yaw [radians]\n");
```

```
    for (i = 0; i < 6; i++)
    {
        sum[i] = 0;
    }


    for (i = 0; i < arr_length; i++)
    {
        for(j = 0; j < 3; j++)
        {
            acc_variance[j] = 0;
            gyro_variance[j] = 0;
            gyro_distance[j] = 0;
            velocity[j] = 0;
            distance_traveled[j] = 0;
            previous_velocity[j] = 0;
        }

        // CALCULATE VARIANCE FOR BOTH ACCELOREMETER AND GYROSCOPE
        acc_variance[0] = calc_variance(accX, arr_length, i);
        acc_variance[1] = calc_variance(accY, arr_length, i);
        acc_variance[2] = calc_variance(accZ, arr_length, i);
        gyro_variance[0] = calc_variance(pitch, arr_length, i);
        gyro_variance[1] = calc_variance(roll, arr_length, i);
        gyro_variance[2] = calc_variance(yaw, arr_length, i);

        // IF TRUE, ACCELOREMETER IN MOTION
        if ((acc_variance[0] > ACC_THRESHOLD_X) || (acc_variance[1] >
ACC_THRESHOLD_Y)
            || (acc_variance[2] > ACC_THRESHOLD_Z))
        {
            moving = 1;
        }

        // IF TRUE, GYROSCOPE IN MOTION
        if ((gyro_variance[0] > GYRO_THRESHOLD_PITCH) || (gyro_variance[1] >
GYRO_THRESHOLD_ROLL)
            || (gyro_variance[2] > GYRO_THRESHOLD_YAW))
        {
            moving = 1;
        }

        // CHECKS WHEN MOVEMENT STARTED AND ENDED
        if (start_movement == 0 && moving == 1)
        {
            start_movement = i;
            start_time = start_movement * SAMPLE_TIME;
        }
        if (end_movement == 0 && moving == 0 && start_movement != 0)
        {
            end_movement = i;
            end_time = end_movement * SAMPLE_TIME;
        }

        // ONCE MOVEMENT PERIOD OBTAINED, GYROSCOPE AND ACCELERATION INTEGRATION IS
CALCULATED
        if (start_movement != 0 && end_movement != 0)
        {
            // GYROSCOPE INTEGRATION
            for (j = start_movement; j < end_movement; j++)
            {
                gyro_distance[0] += ((*pitch)[j] * SAMPLE_TIME);
                gyro_distance[1] += ((*roll)[j] * SAMPLE_TIME);
                gyro_distance[2] += ((*yaw)[j] * SAMPLE_TIME);
            }

            // ACCELEROMETER DOUBLE INTEGRATION
            for (j = start_movement; j < end_movement; j++)
            {
                previous_velocity[0] = velocity[0];
                velocity[0] += ((*accX)[j] * GRAVITY * SAMPLE_TIME);
                distance_traveled[0] += (((velocity[0] + previous_velocity[0]) / 2) *
SAMPLE_TIME);

                previous_velocity[1] = velocity[1];
                velocity[1] += ((*accY)[j] * GRAVITY * SAMPLE_TIME);
                distance_traveled[1] += (((velocity[1] + previous_velocity[1]) / 2) *
SAMPLE_TIME);

                previous_velocity[2] = velocity[2];
                velocity[2] += ((*accZ)[j] * GRAVITY * SAMPLE_TIME);
                distance_traveled[2] += (((velocity[2] + previous_velocity[2]) / 2) *
SAMPLE_TIME);

            }

            sum[0] += distance_traveled[0];
            sum[1] += distance_traveled[1];
            sum[2] += distance_traveled[2];
            sum[3] += gyro_distance[0];
            sum[4] += gyro_distance[1];
            sum[5] += gyro_distance[2];

            // TEXT FILE PRINTS
            fprintf(text_file, "-------------------------------------\n");
            fprintf(text_file, "Movement #%d:\n", k);
            fprintf(text_file, "Movement X-axis: %.6f[m]\nMovement Y-axis: %.6f[m]
\nMovement Z-axis: %.6f[m]\n", distance_traveled[0], distance_traveled[1],
distance_traveled[2]);
            fprintf(text_file, "Movement Pitch: %.6f[radians]\nMovement Roll: %.
6f[radians]\nMovement Yaw: %.6f[radians]\n", gyro_distance[0], gyro_distance[1],
gyro_distance[2]);
            fprintf(text_file, "Movement Start Time: %.2f | Movement End Time: %.
2f\n", start_time, end_time);
            fprintf(text_file, "Movement Start Index: %d | Movement End Index: %d\n",
start_movement, end_movement);
            fprintf(text_file, "-------------------------------------\n\n");

            // CSV FILE PRINTS
            fprintf(csv_file, "%d,%d,%.2f,%.2f,%.6f,%.6f,%.6f,%.6f,%.6f,%.6f\n",
                    start_movement + 1, end_movement + 1, start_time, end_time,
                    distance_traveled[0], distance_traveled[1], distance_traveled[2],
                    gyro_distance[0], gyro_distance[1], gyro_distance[2]);

            start_movement = 0;
            end_movement = 0;
            start_time = 0;
            end_time = 0;
            k++;
        }
        moving = 0;
    }
    fprintf(csv_file,"Total Distance:,,,,%.6f,%.6f,%.6f,%.6f,%.6f,%.6f\n",
```

```c
sum[0],sum[1],sum[2],sum[3],sum[4],sum[5]);
    fclose(text_file);
    fclose(csv_file);
}

int main(int argc, char *argv[])
{
    /* VARIABLE DECLARATION SECTION */
    FILE *file;
    int file_size;
    int i;
    double *time, *accX, *accY, *accZ, *pitch, *roll, *yaw;
    double *smooth_accX, *smooth_accY, *smooth_accZ, *smooth_pitch, *smooth_roll,
*smooth_yaw;

    if (argc != 2)
    {
        printf("Usage: ./executable text_file.txt\n");
        exit(1);
    }

    /* EXTRACT DATA FROM TEXT FILE */
    read_text_file(argv[1], &file_size, &time, &accX, &accY, &accZ, &pitch, &roll,
&yaw);

    /* SMOOTH EXTRACTED DATA */
    smooth_data(file_size, &accX, &smooth_accX);
    smooth_data(file_size, &accY, &smooth_accY);
    smooth_data(file_size, &accZ, &smooth_accZ);
    smooth_data(file_size, &pitch, &smooth_pitch);
    smooth_data(file_size, &roll, &smooth_roll);
    smooth_data(file_size, &yaw, &smooth_yaw);

    /* EXPORT SMOOTHED DATA AS CSV */
    file = fopen("smoothed_data.csv", "w");
    fprintf(file, "Time[s],Acceletarion-X[m/s],Acceletarion-Y[m/s],Acceleration-Z[m/
s],Pitch,Roll,Yaw\n");
    for (i = 0; i < file_size; i++)
    {
        fprintf(file, "%lf,%lf,%lf,%lf,%lf,%lf,%lf\n", time[i], smooth_accX[i],
smooth_accY[i], smooth_accZ[i], smooth_pitch[i], smooth_roll[i], smooth_yaw[i]);
    }
    fclose(file);

    /* DETERMINE MOVEMENT */
    move_or_still(&accX, &accY, &accZ, &pitch, &roll, &yaw, file_size);

    return 0;
}
```