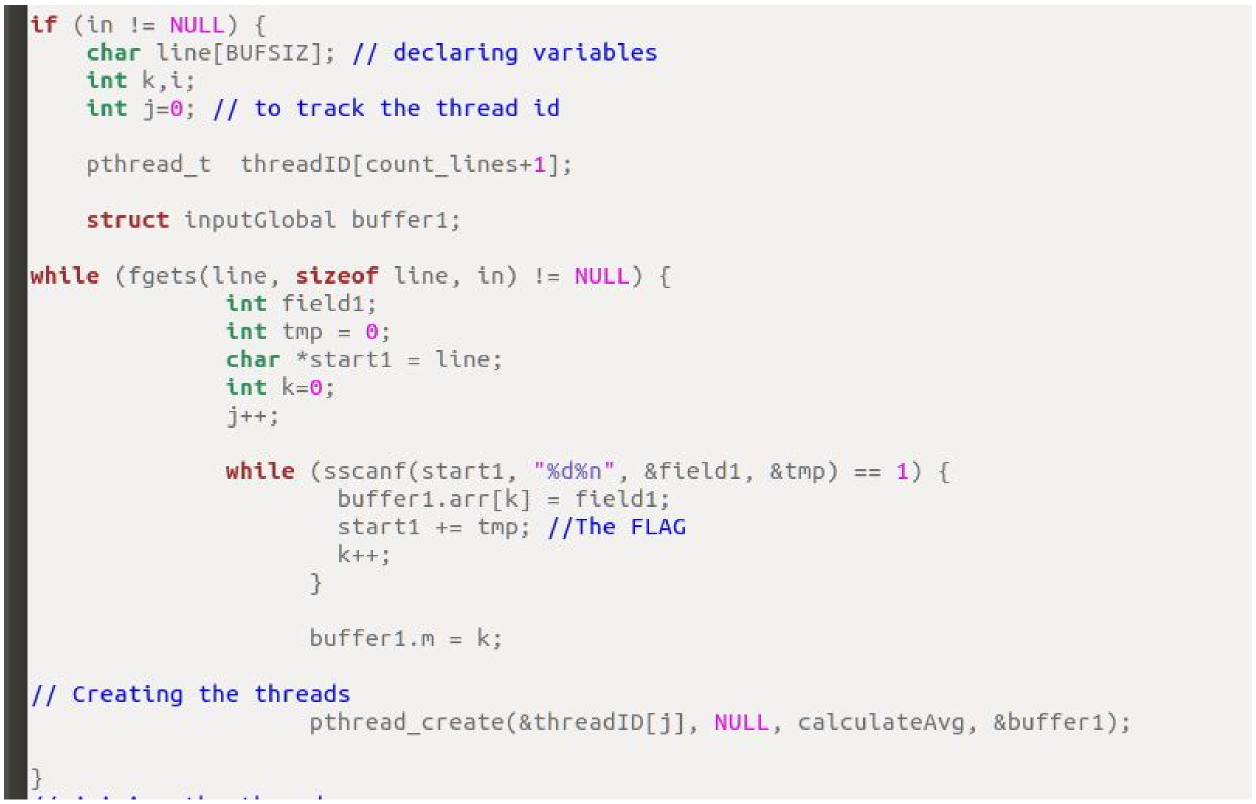
TASK 1 - IT18350906 (I.R Aushan)

The primary obstacle that I came across when attempting to complete the code is to read lines, with variable number of values and passing them into the calculateAvg() function, one set at a time.



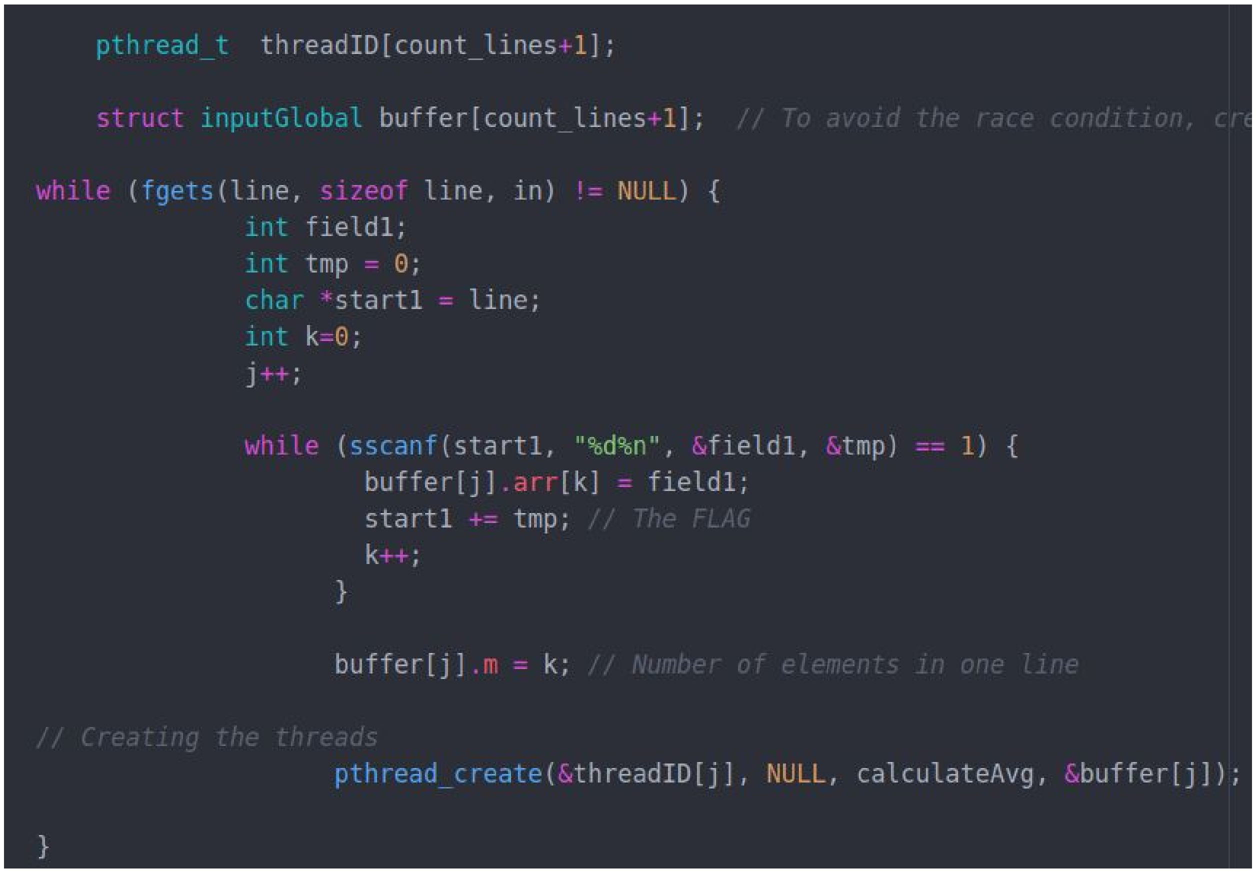
```
if (in != NULL) {
    char line[BUFSIZ]; // declaring variables
    int k,i;
    int j=0; // to track the thread id

    pthread_t  threadID[count_lines+1];

    struct inputGlobal buffer1;

while (fgets(line, sizeof line, in) != NULL) {
              int field1;
              int tmp = 0;
              char *start1 = line;
              int k=0;
              j++;

              while (sscanf(start1, "%d%n", &field1, &tmp) == 1) {
                      buffer1.arr[k] = field1;
                      start1 += tmp; //The FLAG
                      k++;
                    }

                    buffer1.m = k;

// Creating the threads
                    pthread_create(&threadID[j], NULL, calculateAvg, &buffer1);

}
```

In the above code segment, the fgets function reads a line until it meets a newline character, and passes the whole line into the char line array. And sets it's pointer to first character of the next line for next loop cycle.
The line array's first character is pointed by the start1.
The function sscanf() read the line using that pointer,passes the integer value to field and the %n format specifier sends the number of blankspaces to tmp. Then that value (tmp) is added to the start1, so that it skips the blank space and points to the next value. Likewise the loop continues and the field1 integers are passed to the array of the buffer1 structure. Which is then used by the thread.

This method ensures that no thread retrieves the values that already taken by another thread.
k variable is used to counter number of integers per line.

Mutexes are used inside the calculateAvg() function to ensure it doesn't get interrupted. And the Globally declared struct values aren't mixed up while calculating. (Please cont. reading)

But little did I know, using the same structure to pass in all the lines generates another race condition!
To avoid that, I made dynamic structure, Meaning the data is passed to different instances of the globally declared inputGlobal. (see code below)



```
    pthread_t  threadID[count_lines+1];

    struct inputGlobal buffer[count_lines+1];  // To avoid the race condition, cre

while (fgets(line, sizeof line, in) != NULL) {
              int field1;
              int tmp = 0;
              char *start1 = line;
              int k=0;
              j++;

              while (sscanf(start1, "%d%n", &field1, &tmp) == 1) {
                      buffer[j].arr[k] = field1;
                      start1 += tmp; // The FLAG
                      k++;
                    }

                    buffer[j].m = k; // Number of elements in one line

// Creating the threads
                    pthread_create(&threadID[j], NULL, calculateAvg, &buffer[j]);

}
```

References:

LABSHEET 3 to 5
https://www.tutorialspoint.com/c_standard_library/c_function_fgets.htm
https://www.daniweb.com/programming/software-development/code/216411/reading-a-file-line-by-line
https://stackoverflow.com/questions/3401156/what-is-the-use-of-the-n-format-specifier-in-c
http://www.zentut.com/c-tutorial/c-write-text-file
http://softpixel.com/~cwright/programming/threads/threads.c.php
https://cboard.cprogramming.com/c-programming/135530-how-detect-end-line.html
https://cboard.cprogramming.com/cplusplus-programming/46898-read-end-line-problem.html?s=9e0bdbeede10ed9dd8c738ec32712c02
https://www.daniweb.com/programming/software-development/threads/398132/read-integers-with-spaces-from-a-file-into-an-integer-array
https://www.quora.com/How-do-you-read-integers-from-a-file-in-C

https://www.javatpoint.com/file-handling-in-c
https://www.cs.utah.edu/~germain/PPS/Topics/C_Language/file_IO.html
https://computing.llnl.gov/tutorials/pthreads/