Assignment 2 – Code Explanation & Documentation

This C program will compute the sum of square roots from 1 to n, where n is a multiple of 3, and n is specified in the command line. The program gets the value n from the command line argument (using argc / argv) and converts it to an integer after checking that it is in the right format. Next, it creates two threads using pthread\_create(). Then, the main thread computes the sum of square roots from 1 to n/3, while the two child threads compute the sum of the square roots from n/3+1 to 2n/3 and 2n/3+1 to n, respectively, at the same time as the main thread’s calculation using the \*sum\_of\_sqrt function call. All three partial sums are added together to a shared global variable (grand\_sum) by each thread at the end of execution. The main thread calls pthread\_join() to wait for the termination of the two child threads and then prints the final result.

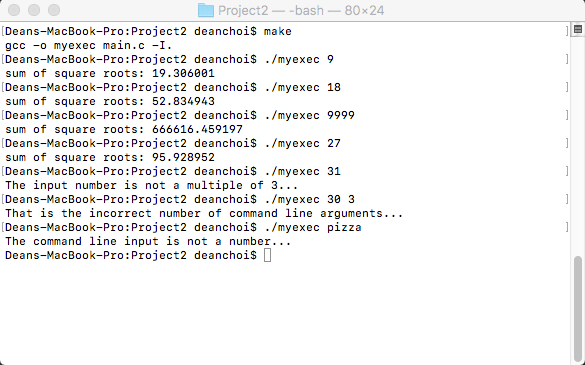
The source code of this program is in the file named "main.c". It comes with an accompanying Makefile in the project folder.

To compile the program just type in the “*make*” command at the Terminal. To properly run the executable binary, type in “.*/myexec*” followed by an argument for the number n, which is a multiple of 3. Thus, the complete execution command is *“./myexec n”* where n is an integer that is a multiple of 3.

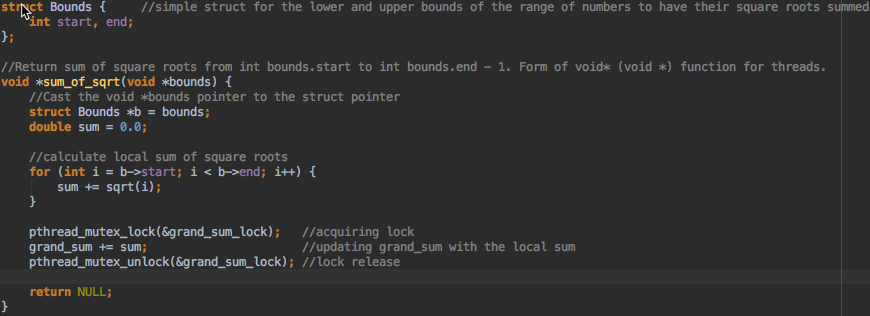
If the Makefile is not present, you can compile the executable via gcc in Terminal with the following command:

*gcc -o myexec main.c -I.*

The program will determine if the correct command line arguments have been entered. If not, the corresponding error will be displayed. This is a sample output below:

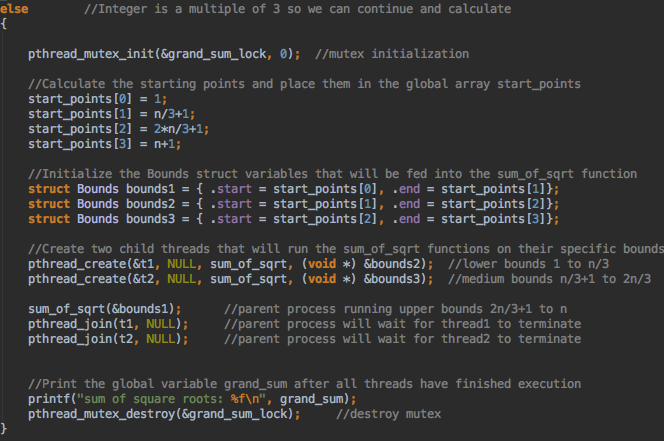


Our code is well-documented with comments within the source main.c file. However, we will expand upon the code a little bit in the documentation as well.



We first had to define a function called **sum\_of\_sqrt** that would locally calculate the sum of square roots for a certain range of numbers that were indicated. In order to more easily pass in those bounds parameters, we created a struct called “**Bounds**” that had two integer values for the **start** and **end** of the bounds that we wanted to calculate the square root sums for.

Our “**void \*sum\_of\_sqrt(void \*bounds)**” function would take in a **void \*** pointer as is required for pthread\_create() functions that would be cast into the proper **struct Bounds** pointer within. Then, we could locally calculate the sum of square roots from the bounds->start to the bounds->end value. Afterward, the function would access the mutex lock to see if it was free and if the critical region was clear, it would then lock the mutex, add that local sum to the grand\_sum global variable, and then finally unlock the mutex once it was all done.



The two children threads created via the pthread\_create calls would calculate the square root sums of the middle and upper bounds, while the main parent process would calculate the square root sum of the lower set of bounds. These were all added to the global\_sum variable.

The parent thread would wait for the two children threads to terminate via the pthread\_join call, and only then, would it then print out the total sum of the square roots that was calculated in the global\_sum variable. Afterward, the mutex would be destroyed.