Executive summary/overview:

This code implements a program that reads star data from a file, calculates the angular distance between each pair of stars, and determines the minimum, maximum, and average distance between them. The program is designed to use multiple threads to speed up the calculations. The program then calculates the angular distance between each pair of stars by iterating through two nested loops. Each thread calculates the distance for a portion of the star pairs. The calculation of the distance is protected with a mutex to avoid race conditions. Finally, the program determines the minimum, maximum, and average distance between each pair of stars and outputs the results.

Libraries:

- time.h: This library provides functions to work with time, including measuring elapsed time.
- math.h: This library provides mathematical functions that are used in the code, such as square root.
- stdlib.h: This library provides general-purpose functions such as memory allocation and program termination.
- stdio.h: This library provides functions for input/output operations, such as printf and scanf.
- string.h: This library provides functions for manipulating strings, such as strcpy and strlen.
- assert.h: This library provides a macro for debugging purposes, which is used in the code.
- getopt.h: This library provides functions to parse command-line arguments.
- stdint.h: This library provides fixed-size integer types that are used in the code.
- pthread.h: This library provides functions for creating and manipulating threads.
- utility.h: This is a custom header file that contains utility functions used in the code.
- star.h: This is a custom header file that contains the definition of the Star struct used in the code.
- float.h: This library provides information about floating-point arithmetic, such as the minimum and maximum values that can be represented.

The time.h library is used in the code to measure the execution time of different parts of the program. The clock() function is used to measure the CPU time used by the program. The CLOCKS_PER_SEC constant is used to convert clock ticks to seconds.

The **pthread** library is used in the code to create and manage threads to parallelize the computation of distances between stars in the **compute_distances()** function. The **pthread_create()** function is used to create a new thread for each range. The **pthread_join()** function is used to wait for each thread to finish computing distances before collecting the results and continuing with the rest of the program.

By using multiple threads, the **compute_distances()** function can be executed in parallel, which can significantly improve performance on multi-core systems.

Discussion: Non-deterministic performance on the cloud provider. The shared compute resources don't provide a consistent performance. I had an anomaly in which the time results when using 2 threads was significantly higher than any other number of threads. Furthermore, I had numerous segfaults involved in my calculation.

Conclusion:

In general, using a moderate number of threads (e.g. 1-10) provides a good balance between parallelization and overhead with the best time occurring with 2 threads. Using a high number of threads resulted in very high overhead due to the large number of threads and the potential for contention for shared resources, and resulted performance compared to using fewer threads.

Number of Threads	Time (Seconds)
1	58.57
2	58.22
4	58.48
10	58.36
25	197.89
100	424.61
1000	499.92