CLE - Assignment 1

Eduardo Santos - 93107
Pedro Bastos - 93150

Problem 1

main.c - The objective of the main thread is to generate the worker threads, waiting for their termination and printing the final results.

methods:

- **main(int argc, char *argv[])** - handles the command line options to retrieve the files, starts the worker threads, waits for their termination and prints the final results.
- **worker(void *par)** - each worker thread will execute to compute the file chunks.
- **printUsage(char *cmdName)** - print the command line usage, if needed.

The worker keeps on going while there is files to read (using the **check_for_file** method in the shared region) and while there is chunks to process of the current file (using the **getVal** method in the shared region).

Problem 1

shared.c - This is the file that contains the shared region between the different threads. Basically, its objective is to read the files, process each chunk, save the results and close the files.

methods:

- **check_for_file(unsigned int id)** - open each file, if available, making sure it is opened only once. It waits until all the workers are ready to read the next file using a condition.
- **check_close_file(unsigned int id)** - close each file, making sure it is closed only once. It waits until all the workers are ready to close the file, using a condition.
- **getVal(unsigned int consId)** - processes each chunk and updates the counting variables. The methods **get_int**, **is_vowel**, **is_consonant** and **is_split** are used by this to compute the chunk.
- **save_file_results(consId)** - saves the results of each file.
- **print_final_results()** - Carried out by the main thread. Its role is to print the final results.

Problem 2

main.c methods:

- main() - where the worker threads are created.
- computeDet(int order, double **matrix) - method to compute the determinant.
- process_comand(int argc, char *argv[]) and printUsage (char *cmdName) - where the program usage is defined.
- *worker (void *par) - what each worker thread will execute to compute the determinants of the matrices.

Problem 2

sharedRegion.c methods:

- openNextFile() - opens next file and processes it.
- storeFileNames(int filesNumber, char * fileNames[]) - store given filenames in new array, to be opened.
- getVal(int threadID, PartialInfo *info) - Get a value from the data transfer region.
- savePartialResults(int threadID, int fileID, int matrixNumber, double det) - save partial results

PartialInfo.c: Struct to be used on the shared region with data for each matrix.

Execution Times

Problem 1

The threads successfully decrease the execution time.

	1 thread	2 threads	4 threads	8 threads
Execution 1	0.54293 s	0.303768 s	0.155398 s	0.079124 s
Execution 2	0.534681 s	0.299032 s	0.148467 s	0.075362 s
Execution 3	0.552529 s	0.292554 s	0.144136 s	0.073569 s

Problem 2

As we can see, the execution times decrease as the number of threads increases, which tells us that our program seems efficient, and that it complies with the objective of the assignment.

File	# of threads	Execution time
mat128_32.bin	1	0.007675 s
mat128_32.bin	2	0.002463 s
mat128_32.bin	4	0.002393 s
mat128_32.bin	8	0.002480 s
mat128_256.bin	1	0.462604 s
mat128_256.bin	2	0.445569 s
mat128_256.bin	4	0.433968 s
mat128_256.bin	8	0.429465 s
mat512_32.bin	1	0.010445 s
mat512_32.bin	2	0.010290 s
mat512_32.bin	4	0.010886 s
mat512_32.bin	8	0.008761 s
mat512_256.bin	1	1.839120 s
mat512_256.bin	2	1.776207 s
mat512_256.bin	4	1.684873 s
mat512_256.bin	8	1.672373 s