Lab 02 – Mensio Martino

Exercise 1

For this exercise it was required to replace the recursive calls to MergeSort with thread activations. For each step of the recursion I decided to activate a Left thread and a Right thread, each of which is devoted to processing his part of the array. Then it waits to join with this two threads. In this way there is no need to use semaphores, because each thread accesses a different half of the (sub)array. The wait of the two generated threads at each step of recursion before calling the merge function, assures independence of threads in matter of data and also the correctness of the results.

Each thread receives a parameter of type mergeParamType, that contains all the information needed (the pointer to the array and the extremes of the portion it has to manage.

Exercise 2

For the second exercise, I introduced some modifications to the code in order to stop the creation of the threads if the size of (sub)array is lower than a specified threshold. I created a function that sorts a subarray with a bubble sort, and is called when the size of the array is lower than the threshold. I simply could have used the same function that is executed by threads without creating a new thread instead of writing another function.

The parameter structure is extended in order to propagate the value of the threshold to successive recursions.

Exercise 3

The third lab required to compute some matricial products. In order to generalize, in the "serial" implementation (single-threaded), I wrote some functions to manage properly the arrays and matrices. The representation I chose to use is the matrix one, also for arrays. I have one function to transform an array of a given size k into a matrix of dimension (k, 1).

The products are computed using a general function that receives two matrices ((m, n) and (n, o)) and returns a matrix (m, o).

Instead in the multi-threaded version, since the requirements were specific about the job that each thread needed to do, I couldn't use some "general" approach.

In order to have the last thread doing the final operation, I used a global variable protected by a mutex. In this way, only the last thread will find a specific value in it and will be able to understand to be the last one.