We wish to find the largest element in a list of integers. This will be done in two steps.

- In the first step, a parent process will create k child processes. Each child process will find the largest element in the sub-list assigned to it and write this number in the aprropriate location of a result list.
- When the first step is completed by all child processes, the parent process will find the largest element in the result list filled in by the child processes.

The parent process will output the largest element it finds in the list.

Example:

- A list L, in which the largest element will be found, has 24 elements as given below. Assume that the parent process creates 4 child processes. The child processes will write the largest elements they find in the assigned sub-lists to result list RL.

The element count for list RL depends on the number of child processes.

- In the example, there are 4 child processes, so RL list has 4 elements.

The number of elements in the sub-lists assigned to the child processes depends on the total number of elements in list L and the number of child processes. (Note: All sub-lists should contain the same number of elements, i.e. the total number of elements in list L should be a multiple of the number of child processes).

- In the example, the total number of elements in list L is 24 and there are 4 child processes. So, each sub-list contains 6 elements.

L[24]: [1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24]
Child 0: [[1 2 3 4 5 6] → RL[0] = 6
Child 1: [7 8 9 10 11 12] → RL[1] = 12
Child 2: [13 14 15 16 17 18] → RL[2] = 18
Child 3: [19 20 21 22 23 24] → RL[3] = 24
EB[4]: [6 12 18 24]
Parent Process finds the largest element as 24.

Implementation: A parent process will create a list L with n elements in a shared memory area and fill the list with random integers. The parent process will also write the number of elements in the list (n) and the number of child processes (k) in another shared memory location. The results list RL, where the largest elements of the sub-lists are stored will also be created by the parent process as another shared memory location. n and k will be provided as command line parameters. Note that n must be a multiple of k.
The parent process will then create k child processes and wait until all child processes find the largest elements in their corresponding sub-lists and write them into the appropriate locations in RL. Each child process will calculate the starting index and the number of elements of its sub-list based on the order in which the child was created (please see example above).
When all child processes are finished, the parent process will find the largest element in list RL and print it. All resources will be given back to the system at the end by the parent process.

What to do: Write the code for the problem explained above. Use the appropriate system calls and IPC primitives.
(a) Use processes.
(b) Use threads instead of processes. (i.e. for the example above there will be 5 threads).