

Communication and Information Engineering

Parallel and Distributed Computing (CIE564)

Instructors: **Dr. Anas Youssef**

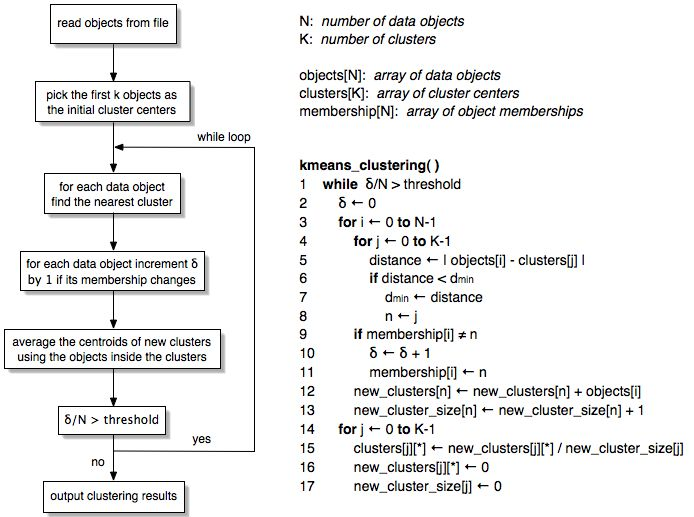
TAs: **Hanan Khaled** and **Reem Abdelsalam**

**Assignment 2**

**Student should submit a copy of his/her source codes in one zip file on the Google classroom assignment submission link on or before 23:59 Thursday, Nov 26, 2020 – late submission is not allowed**

**Question 1: (5 marks)**

1. Implement the **K-means algorithm shown in Figure 1** to cluster data points. The test cases consist of 2 input text files (attached test1.txt and test2.txt). The first line in each of the two files contains the number of points while the following lines contain the co-ordinates of each point. Use **OpenMP** to parallelize the K-means algorithm. There will be no code skeleton provided for this part. The serial algorithm is shown in Figure 1.



**Figure 1**

1. Determine the performance of your implementation of the K-means algorithm. Be careful when taking times in your code, you should calculate only the time after processing the input. Use *omp\_get\_wtime()* function to take times for you implementation. Print the time taken before the program ends. Report the performance for the two test cases.

**Question 2: (5 marks)**

1. Implement the **MergeSort Algorithm** to sort an array of size 10K with elements of datatype ‘int’. The array must be initialized using a random number generator. Use OpenMP to parallelize the MergeSort algorithm. If statically allocating memory results in segmentation fault, use dynamic memory allocation for the array. There will be no code skeleton provided for this part.

**HINTs**: You may need OpenMP (OMP) directives such as: *omp task* and *omp taskwait.* You may also need OMP clauses such as: *shared* and *firstprivate.*

1. Determine the performance of both your serial and your parallel implementations of the MergeSort Algorithm.