# COEN346

# Programming Assignment 1

# Report

Dr. Ferhat Khendek

Section F

Nabila Tabassum ID:40109039

Jaskirat Kaur ID: 40138320

Laura Hang ID: 40203006

## Introduction:

In this project, we created a program that gets integer values from the input file and sorts them using a recursive merge sort algorithm. Further, the output is redirected to a new file and the list of sorted integers. This program is written in the python programming language. We implemented it using threads, allowing our program to operate more effectively by sorting in parallel. In the output file, output clearing states the starting /stopping of each thread as well as the sorted sub-list corresponding to each thread.

## Description of the functions:

**def divide_the_arraylist(array: list):**

This function divides the list into two sub-list. First, we find the index in the middle of the list. Then stores the values from left to the middle are in one sub-array list and from the middle index to the right are stored in another sub-array.

**def merge_sort(i: list, j: list):**

We implemented the merge sort in this function on both the left and right arrays. The merge sort algorithm recursively divides the array into halves until we reach the base case of an array with one element. After that, the merge function picks up the sorted sub-arrays and merges them to gradually sort the entire array. This merged array is stored in 'merged_arr'.

**def threaded_mergesort (array: list, sorted_arr: list)**

This function defines the custom thread name and keeps track of starting and joining of the threads. We used 2 threads one for each half (sub-array of left and right). Both threads start simultaneously, so both sides are parallelly sorted. After that, the threads are joined, and the sorted array is stored in the 'sorted_arr'.

**def call_threaded_mergesort(array: list):**

The Output.txt file, which was opened in the main function and is used to print the desired output according to the requirements as stated in the assignment description. It states the starting and finishing of each thread corresponding to the subarray list. Then the threaded_mergesort(array, sorted_arr) function is called and returns results.

In the main function, we first open our input file. Then, take all the inputs from the file and save them in an array using the append function. An output text file is also opened. Finally, we call the threaded_mergesort(array) to print the required output in the output file.

## Program flow:

First, our program opens input.txt in reading mode. Then copy each element into an array as integers. An array is then divided into two sub-array – left array (denoted as i) and right array (denoted as j). After that our merge sort algorithm comes to play. Each sub-array has its own threads. These threads start and both sides are sorted in parallel. When the list is sorted, threads are joined. Then the main output.txt is opened and the threaded merge sort function is called. Finally, the desired output is printed on output.txt.

## Conclusions:

We were able to implement all that is required for this assignment. We used multithreading to improve the performance of our code. We implemented threads and learned how they can provide parallelism within a process. It makes the processes efficient in time and memory. The only issue we got was that with more threads, code becomes difficult to debug and maintain.

In conclusion, this programming assignment helps us troubleshoot, work in a team and be able to separate the sections between the three of us. The work was divided fairly, each of the team members focused on the codes, helping each other when needed and when one was unable to continue because of bugs or confusion. Finally, we got to learn the basics of threads and processes through this assignment.