# COSC 336

# Assignment 02

Instructions: Turn in a PDF report with images/screenshots of your work in detail.

Submission: Include everything in a folder, zip and name it <HW02_Last_First>.zip and submit to Blackboard.

## Part A

Analyze the following recurrences and show their time complexity functions using (1) iteration method and (2) Master Theorem.

A1. $T(n) = 2T\left(\frac{n}{4}\right) + 3$

A2. $T(n) = 3T\left(\frac{n}{4}\right) + 2n$

A3. $T(n) = T(n-2) + 3$

A4. $T(n) = 2T(n-1) + 1$

A5. $T(n) = 4T\left(\frac{n}{2}\right) + n\log n$

A6. $T(n) = 3T\left(\frac{n}{5}\right) + n\log n$

A7. $T(n) = 2T\left(\frac{n}{3}\right) + n^2$

## Part B

Do 2.3-4 (p39) and Problem 2-1 (p39)

## Part C

Implement MERGE-SORT() algorithm that reads from a file named "inputHW02.txt" a list of double numbers (max = 3,000,000 numbers), sorts those numbers and indicates time consumption. This programming question will address the advantage of using iteration loops over recursive calls as well as using INSERTION-SORT() as a procedure in MERGE-SORT().

Your program must perform the following actions:

1. Opens the given file name and reads all double numbers. For simplicity, we assume this file only contains numbers and nothing else.
2. Implements the function INSERTION-SORT() that only sort an array of maximum 25 numbers. The idea is that INSERTION-SORT() will be used as a sub-procedure to sort any sub-array when its size is small enough.
3. Four versions of MERGE-SORT() namely
    a. MERGE-SORT-A(): Using recursive calls and NO INSERTION-SORT() as a sub-procedure
    b. MERGE-SORT-B(): Using ITERATIVE loops (i.e, NO recursion) and NO INSERTION-SORT() as a sub-procedure.
    c. MERGE-SORT-C(): Using recursive calls and INSERTION-SORT() as a sub-procedure.
    d. MERGE-SORT-D(): Using ITERATIVE loops (i.e, NO recursion) and INSERTION-SORT() as a sub-procedure.

4. For testing purpose, write another procedure to randomly generate `N` numbers and write them to a given file name `filename` where `N` and `filename` are input parameters.

Report your MERGE-SORT() time consumption on the following input sizes: `N = 1M, 1.5M, 2M, 2.5M` and `3M` numbers. That is, for each input size N, calls step 4 to generate N numbers and write them to the given file name (e.g., "inputHW02.txt"). Then reads from that file and performs MERGE-SORT-A(), MERGE-SORT-B(), MERGE-SORT-C(), MERGE-SORT-D() and report each of their time consumptions in a graph.