CS 2420  
Sort

# Purpose to familiarize you with

* **Sorting Algorithms**
* **Recursion**
* **Analysis**

## Purpose

The purpose of this lab is to demonstrate the speed difference between Quick-Sort and Merge-Sort verses Selection-Sort and Insertion-Sort. This difference is not shown if the array size is too small. Therefore you should make your arrays as large as possible (without having it take too long), and still have 2 significant digits of output. Quick and Merge are much faster than insertion and selection. Therefore you may need to put quick and merge into a loop and divide the result by the number of iterations. You should also re-randomize each time to give fair results to each sort.

# Tasks

* Create a Sort Class
  + Member variables
    - Dynamic array of int
    - Size of the array
  + Constructor(array size)
  + Destructor
  + Overloaded << operator
    - friend ostream& operator<< (ostream& out, const Sort& s)
    - for testing
    - To display your array.
    - (you don’t really need this but I want you to have practice with this operator)
  + Insertion Sort 🡪 *You need to write all of this.*
  + Selection Sort 🡪 *In book pg. 634 – pay attention to the* swap() *function you need to write.*
  + Merge Sort 🡪 *in book pg. 646; 650 – you just need to copy this. (or write your own )*
  + Quick Sort 🡪 *In book pg. 656 – you still need to write* partition()*.*
    - QuickSort and partition should be separate functions.
      * With quicksort calling partition
      * Also you need to follow the pseudo-code from the book
  + Function to initialize the array with random numbers roughly the same size as the array
    - So if the array was size 20 the random numbers should be between 0-20
  + You will also need several helper functions.
    - These should be private.
* Create a non-interactive driver that runs each of these sorts and times and displays the outputs.
  + To do one sort of array size: ‘size’.
  + You will need a large array size and perhaps a loop.
    - Your results should have at least 2 significant digits
  + Display the output to do one sort in milliseconds
    - For each of the 4 sorts.
    - You may need to do an appropriate division or multiplication.
      * Figuring out how to do this is part of the assignment.

# Notes

#### Random Numbers

* Random Number generation in C/C++ is defined in the <random> header file.
* Use void “srand(long)” to set the random number seed. The seed can be any value, but should be the same each time (so you can duplicate your results).
* Use “int rand()” to generate a random number from 0 to MAXINT.
* Use the % (mod) operator to shrink the random numbers to something more reasonable – like 0-10000 (depending on your array size).

#### Timing

* Timing in C/C++ is defined in the <ctime> header file
* Elapsed time is measured in the clock\_t type.
* Elapsed time to this point in clock ticks is returned by ctime()
* The number of clock ticks per second is defined by a constant defined in ctime called CLOCKS\_PER\_SEC
* Sample Code:

clock\_t start = clock();

// some code to time here

clock\_t end = clock();

clock\_t diff = end - start;

# Turn in

* Code (.h and .cpp files)
* Windows executable