**Notes:**

* You are required to upload your in-class implementations of problems 1-2 to canvas. This is due by 9:50 AM today.
* You are required to turn in a written report (Word or PDF file) for the homework part (problems 3-7) of the lab and upload implementations to canvas. These are due by 8:00 AM, October 4, 2017.
* Homework report must follow the guidelines provided earlier.

**Objectives:**

* Implement basic quick sort algorithm
* Implement quick sort using median of 3 partitioning
* Compare the performance of insertion sort, merge sort, heap sort, and quick sort.

**Problems**

1. Implement a method to sort a given array using the basic quicksort algorithm. Use the algorithm from the textbook (see page 2)
2. Write a driver program to test the quicksort algorithm for the file uploaded in the canvas.
3. Compare the performance of the quicksort algorithm with 3 cases of input files: sorted, reversed sorted, and random. These files are provided in Canvas in the Quicksort Input Files folder.
4. Use the median of 3 partitioning algorithm (given in the next page) to implement quick sort. This algorithm chooses the pivot element as the median of the 3 elements namely: A[p], A[r], and A[(p+r)/2].
5. Compare the performance of the quicksort using median of 3 partitioning with the basic quicksort algorithm using the input files located on Canvas in Quicksort Input Files folder.
6. Compare the execution time of quicksort with the execution time of insertion sort, merge sort, heap sort. Make sure you use the same array to compare the performance. Use a table or plot to summarize the results and document your observations and analysis in the report. You can use the input files ranging from 100 to 1,000,000 or the input files ranging from 16 to 8192.

(See Problem #7 on the following page).

1. Determine the time complexity *T(n)* for the given pseudo code:

i = n

while ( i >= 1) {

j = i

while ( j <= n) {

// some work here

j = j \* 2

}

i = i / 2

}

For simplicity you may assume that n is a power of 2, *i.e.,* n = 2k, for some k > 0.

