Project 5. Deadline: February 11th, 11:59pm.
80 points.

This project has two parts: theoretical and practical.

Point distribution:

Theoretical (30)

- Problem 1: 5 points
- Problem 7.18: 15 points.
- Problem 3: 10 points

Practical(50)

- Insertion sort: 10 points
- Merge Sort: 10 points
- Quick Sort: 10 points
- My Tables: 10 points
- Your own table: 10 points

Part 1. Theoretical Part.

Create HW5_1.pdf file and write all your answers there

Problem 1. For a given array A = (1, 3, 5, 7, 9, 2, 4, 8) and a partition method **below**, show the resulting array after making a call to Partition(A, 0, 7). Show each iteration step.

```
PARTITION(A, p, r)
1  x = A[r]
2  i = p − 1
3  for j = p to r − 1
4      if A[j] ≤ x
5          i = i + 1
6          exchange A[i] with A[j]
7  exchange A[i + 1] with A[r]
8  return i + 1
```

Problem 2: Problem 7.18 from our online textbook
http://people.cs.vt.edu/~shaffer/Book/JAVA3elatest.pdf

Problem 3.

What is the output for printInt(5) and printInt2(5)?

```
void printInt(int k) {
if (k <= 0) return;
System.out.println(k);
printInt(k - 1);
}
```

```
void printInt2 (int k) {
    if (k <= 0) return;
    printInt2(k-1);
  System.out.println(k);
}
```

Part 2. Programming assignment

The best way to understand sorting is to create your own methods. I picked three algorithms that are used in practice the most and would like you to perform a series of experiments.

Step 1. Implement Insertion, Merge, and **Randomized** Quick-Sort (**merge and quick-sort must be *recursive***) using arrays. You can use the method from problem 1 to partition your array for quick sort. You are ALLOWED to use google to help you (site the source) but be aware that I will test your ability to write sorting algorithms in Java. Copying and pasting is OK for this part.

Step 2. Create your test arrays (inputs) and use them for your benchmark testing (use the generated inputs for all sorting methods):

- Random
- Reverse
- "Almost sorted" (at least 80% of values are in correct places). You **may** discuss it with your classmates how to create an almost sorted array.

Common settings for each benchmark:

- Average each test over 100 runs. (time is in nanoseconds)

- 1st test has arrays of size 1,000
- 2nd test has arrays of size 3,000
- 3rd test has arrays of size 200,000
- 4th test has arrays of size 400,000
- 5th test has arrays of size 600,000

Step 3. I wrote the same 3 algorithms and used the same test arrays (same size, same number of runs etc) but forgot to record what algorithm corresponds to what input. :(Once you have your data ready, I'd like you to help me find the missing algorithms by completing the table. Make sure to put the table in your writeup

Almost Sorted	200,000	400,000	600,000	1,000	3,000
??	198721464	1218989888	1777066985	61336	154256
??	16478014	31742470	49281319	325019	397461
??	7450516	15245987	23388518	115733	284131

Random	200,000	400,000	600,000	1,000	3,000
??	4266019882	790915293	1811878853	344227	1822120
??	9381902	15185512	22138816	120993	246266
??	16424205	32710207	51543301	240889	307182

Reverse	200,000	400,000	600,000	1,000	3,000
??	15747693	32501834	53200042	197125	320482

??	6509510	15941183	24013334	106167	251138
??	197390868	788037340	1811804722	64529	136086

Step 4. Create the same tables with your own running times.

What files to submit:

1. HW5_1.pdf
2. HW5_2.pdf with your guesses.
3. HW5_3.pdf with your data.
4. Your java code for the sorting algorithms.