COMP 203 Data Structures and Algorithms Fall 2020
TA : Gökhan Göy
Lab - 4

1) In this problem, you will explore and compare the running times of two sorting algorithms. You can use the following lines to compute the time elapsed for executing each algorithm

```
1 long startTime = System.nanoTime( ); // record the starting time in nanoseconds
2 /* (run the algorithm) */
…
3 long endTime = System.nanoTime( ); // record the ending time in nanoseconds
4 long elapsed = endTime − startTime; // compute the elapsed time
```

(a) Randomly generate arrays of integers with lengths 16, 64, 256, 1024, 4096 and use the Insertion sort method provided in this assignment to sort them. Report the running times of the Insertion sort with respect to the array size. What happens to the running time if you increase the array size by a multiple of 4? What is the big-O complexity of insertion sort and why?

Output should be like this but probably not the same values:

```
Start Time: 146357210942300 ns, End Time: 146357210946900 ns, Elapsed Time: 4600 ns.
Start Time: 146357211936800 ns, End Time: 146357211943600 ns, Elapsed Time: 6800 ns.
Start Time: 146357212184900 ns, End Time: 146357212199200 ns, Elapsed Time: 14300 ns.
Start Time: 146357212445800 ns, End Time: 146357212466200 ns, Elapsed Time: 20400 ns.
Start Time: 146357212713200 ns, End Time: 146357212756100 ns, Elapsed Time: 42900 ns.
```

(b) Sort **the random numbers you generated in part (a)** using the sort() method of Arrays class. Report the running times with respect to the array size. What happens to the running time if you increase the array size by a multiple of 4?

2) Let $p(x)$ be a polynomial of degree n, that is

$$p(x) = \sum_{i=0}^{n} a_i x^i$$

where a_i are the coefficients of the polynomial. You can store these coefficients in an array of size n. Implement a power method in Java for computing x^i as

$$x^i = x * \ldots * x$$

where x is multiplied by itself i times. Then use this method to implement another method that computes $p(x)$. You can use the method declarations given below. What is the big-O complexity of this algorithm and why?

```
public static double polynomial(double a[], double x);
public static double power(double x, int exp);
```