# Exercise 1

Work Individually and submit this document to the LMS with your answers.

1. Arrange the following expressions by growth rate from **slowest to fastest**. (Remember that in each expression, n is the size of the input to the function. You could chart this in excel for different values of n to figure out the answer.)

4n^2 log3n n! 3^n  20n 2 log2n n^2/3

2 log3n log2n n^2/3 20n 4n^2 n! 3^n

1. In Java or C#, implement an algorithm that takes two **sorted** arrays of numbers and returns an array containing the matching numbers. For example, for the input arrays of a1 = [2, 5, 5, 5] and a2 = [2, 2, 3, 5, 5, 7], the output should be [2, 5, 5]. (note, in this case, n = a1.length + a2.length, or n=10)
   1. How long does your algorithm take to finish with the following values for n? (Yes, test this with your algorithm and record the runtime. Lists of random numbers are fine.)
      * 10,000 <1s
      * 100,000 <1s
      * 250,000 <1s
      * 500,000 <1s
   2. What is the maximum number of comparisons/operations your algorithm makes for the above input sizes? Unsure
   3. Given more time, how would you try to optimize your solution? Well considering each of the above each concluded instantly either I’m missing something or it’s already efficient so I don’t think I’d do anything.
   4. Paste your algorithm below.
2. static void Main(string[] args)
3. {
4. Console.WriteLine(DateTime.Now);
5. Random rnd = new Random();
6. int[] one = new int[250000];
7. for(int i = 0; i < one.Length; i++)
8. {
9. int num = rnd.Next(100);
10. one[i] = num;
11. }
12. Array.Sort(one);
13. int[] two = new int[250000];
14. for (int i = 0; i < two.Length; i++)
15. {
16. int num = rnd.Next(100);
17. two[i] = num;
18. }
19. Array.Sort(two);
20. List<int> intersectArr = arraysCommonalities(one, two);
21. Console.WriteLine(DateTime.Now);
22. }
23. //0,0,0,0
24. private static List<int> arraysCommonalities(int[] one, int[] two)
25. {
26. List<int> intersect = new List<int>();
27. int i = 0, j = 0;
28. while (i < one.Length && j < two.Length)
29. {
30. if (one[i] < two[j])
31. i++;
32. else if (two[j] < one[i])
33. j++;
34. else
35. {
36. intersect.Add(two[j++]);
37. i++;
38. }
39. }
40. return intersect;
41. }
42. }
43. In Java or C#, implement an algorithm to find the min, max, and average of numbers in an array using only a single iteration through the array. You may not use any libraries or helper functions. Paste your algorithm below.

int[] numbers = { 8, 3, 7, 4, 11, 4, 4, 66, 10, 3, 2 };

int max = 0;

int min = int.MaxValue;

int sum = 0;

float avg;

for(int i = 0; i < numbers.Length; i++)

{

if (numbers[i] > max)

{

max = numbers[i];

}

if (numbers[i] < min)

{

min = numbers[i];

}

sum += numbers[i];

}

avg = sum / numbers.Length;

1. **Essay Question:** Research Asymptotic Analysis. Explain to me in your own words what it is and why it’s important when thinking about Algorithms. Be specific. 2-5 good paragraphs should be enough but include as much detail as you can. *(Include definitions Big O, Omega and Theta notations in your answer.)*

Asymptotic analysis is the input bound way to determine what an algorithm’s best/average/worst case scenarios of its run-time performance are. There three commonly used asymptotic notions include Big oh notation, omega notation and theta notation.

Big oh- The notation Ο(n) is the formal way to express the upper bound of an algorithm's running time. It measures the worst-case time complexity or the longest amount of time an algorithm can possibly take to complete.

Omega- The notation Ω(n) is the formal way to express the lower bound of an algorithm's running time. It measures the best-case time complexity or the best amount of time an algorithm can possibly take to complete.

Theta- The notation θ(n) is the formal way to express both the lower bound and the upper bound of an algorithm's running time.

More common notations:

