Assignment 1

1. Order the following list of functions by the big-O notation from highest to lowest. Total functions are twelve to arrange.

$n \log n$, $\log \log n$, $1/n$, $4n^{3/2}$, $5n$, $2n \log 2n$,
2^n, 4^n, n^3, $n^2 \log n$, $4^{\log n}$, $n^{1/2}$

Answer:
4^n
2^n
n^3
$n^2 \log n$
$4n^{3/2}$
2n log 2n
n log(n)
5n
$4^{\log n}$
$n^{1/2}$
log log(n)
1/n

2. Give a big-O characterization, in terms of *n*, of the running time of the Loop1 method below:

```
Algorithm Loop1(n)
    s ← 0
    for i ← 1 to n do
        s ← s + i
```

Analysis answer: This is a simple "for loop" with no nested loops and as such has a complexity O(n)

3. Perform a similar analysis for method Loop2 below:

```
Algorithm Loop2(n)
    s ← 0
    for i ← 1 to $n^2$ do
        for j ← 1 to i do
            s ← s + i
```

Analysis answer: this is a "for loop" with a nested loop. Each outer loop traverses n^2 and each inner loop traverses n^2 elements for each "I'. Therefore, the total complexity is $n^2 * n^2$ and results in a total complexity $O(n^4)$.

4. Decide whether each of the following is true or false. Justify your answer why it is true or false.

a. log n is O(n)
b. 2^n is O(n^2)

a) True, since log n grows no faster than n. According to c
b) False, since 2^n grows faster than n^2. Test with sample data and plot it on the graph or can prove using limits.

5. The given code will find the sum of n elements for the array. Your job is to analyse the best and worst case runtime for this given function.

```
function sumArray(arr) {
  if (arr.length === 1) {
    return arr[0];
  }

  let total = 0;
  for (let i = 0; i < arr.length; i++) {
    total += arr[i];
  }
  return total;
}
```

Best Case:

- The best-case scenario occurs when the input array has exactly one element.
- In this case, the algorithm simply checks the length of the array (O(1) time) and returns the single element (O(1) time). So, the best-case time complexity is O(1).

Worst Case:

- The worst-case scenario occurs when the input array has more than one element, requiring a sum of all elements.
- In this case, the algorithm iterates through all the elements in the array (O(n) time), where 'n' is the number of elements in the array. Adding all 'n' elements together has a time complexity of O(n). So, the worst-case time complexity is O(n).

6. Write a Pseudo code to return the count of even numbers in the given array and analyse the Big O runtime for this algorithm.

```
Algorithm arraySum(A)
Input array A of integers
Output sum  of elements of A
evenCount ← 0
for i ← 1 to n  do
    if A[i] is even then
        evenCount ← evenCount + 1
return evenCount
```

Runtime Analysis: The time complexity of this algorithm is O(n), where 'n' is the number of elements in the input array. In the worst case, it need to check each element to determine whether it is even or not.