CS 212 Homework #1

Please complete the homework problems on the following page. Note that this is an individual assignment and all work must be your own. Be sure to show your work when appropriate. This assignment is due **by the beginning of class** on Wednesday, February 6, 2019. Note that **there is no late turn-in** meaning that this assignment must be turned in by the due date.

1. [2] Order the following set of functions by their growth rate:

1. N
2. $\sqrt{N}$
3. $N^{1.5}$
4. N^2
5. $NlogN$
6. $Nlog(\log(N))$
7. $NLog^2N$
8. $\frac{2}{N}$
9. $Nlog(N^2)$
10. 2^N
11. $2^{\frac{N}{2}}$
12. 37
13. N^2logN
14. N^3

2. [3] A program takes 10 seconds for input size 500 (i.e., n=500). Ignoring the effect of constants, approximately how much time can the same program be expected to take if the input size is increased to 2000 given the following run-time complexities?

1. O(N)

2. O(N log N)

3. O(N^3)

4. O(2^n)

3. [2] I've developed a new, super cool sorting algorithm. Here's the procedure:

Until sorted:

1. Randomly generate i = some number between 0 and n - 1 (n = size of array)
2. Randomly generate j = some number between i + 1 and n - 1
3. if array[i] > array[j], swap
4. Check to see if the array has been sorted

What is the best and worst case runtime complexity of this algorithm?

4. [5] State the runtime complexity of the following loops:

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

b.
```
Algorithm Loop2(p):
p ← 1
for i ← 1 to 2n do
    p ← p * i
```

c.
```
Algorithm Loop3(n):
p ← 1
for j ← 1 to n^2 do
    p ← p * i
```

d.
```
Algorithm Loop4(n):
s ← 0
for j ← 1 to 2n do
    for k ← 1 to j do
        s ← s + j
```

e.
```
Algorithm Loop5(n):
k ← 0
for r ← 1 to n^2 do
    for s ← 1 to r do
        k ← k + r
```

5. **Provide the runtime complexity for the following tasks.**

A. [1] Determining whether a provided number is odd or even

B. [1] Determining whether or not a number exists in a list

C. [1] Finding the smallest number in a list

D. [1] Determining whether or not two **unsorted** lists contain all of the same values (assume no duplicate values)

E. [1] Determining whether or not two **sorted** list contain all of the same values (assume no duplicate values)

6. [2] Solve the recurrence relation $T(n) = 4T\left(\frac{n}{2}\right) + n^3$

7. [2] Solve the recurrence relation $T(n) = 3T\left(\frac{n}{2}\right) + nLog(n)$

8. [2] Solve the recurrence relation $T(n) = 3T\left(\frac{n}{3}\right) + \frac{n}{2}$