Exercise 1

One solution is to re-write all to something n^x

1. $2n + lg(9n^3)$:

   $lg(n^3)$ = $3 * lg(n)$ = $lg(n)$

   n vs lg(n), answer: $\Theta(n)$

2. $n * lg^2(n) + \sqrt{n^3}$:

   $n * lg^2(n) = n * (lg(n))^2$

   I guess: $\sqrt{n^3}$

3. $\sqrt[3]{n^2} + n * lg^2(n) + n * lg(n^3)$

   $n * lg^2(n) = n * (lg(n))^2$

   $n * lg^3(n) = n * (lg(n))^3$

4. $2n^{10} + 1.5^n/100 + 4n^9 * lg(n)$

   $\rightarrow n^{10} + xxxx + n^9 * lg(n)$

   Soo: 1.5^n

5. $3^{n/3} + 3^{n/10} + 10^{log_3^n}$

   Of the first two elements: $3^{n/3}$ is worst. The last one $10^{log_3^n}$ can be written as: $n^{log_3^{10}}$ and then its clear:

   answer: $\Theta(10^{log_3^n})$

Exercise 2

Consider an array A[1..n] of points and a function DIST(p1,p2) that computes a distance between any two points p1 and p2 in O(1) time.

2.1 Write an algorithm, that outputs all pairs of points in A that are closer to each other than d units. If two points p_1 and p_2 form a qualifying pair, the two different orderings of these points, (p_1, p_2) and (p_2, p_1), should not be reported as different pairs, the pair should be reported just once.

```
INPUT: d - an integer and A[1..n] - an array of points (p1,p2).
OUTPUT: Array of pairs of points that are closer than d-units, and where no (p1,p2) =
(p2,p1).

KuntzFUNC(A[], d)
result[]
for i = 0 to n-1

    for j = i+1 to n
        dist = DIST(A[i], A[j])
        if dist < d then result.append(pair(A[i], A[j]))

return result
```