1. **(20 points)** Order the following list of functions by the big-O notation.

$6n \log n$	2^{100}	$log\ log\ n$	$log^2 n$	$2^{log\ n}$
2^{2^n}	$\lceil \sqrt{(n)} \rceil$	$n^{0.01}$	$1/n$	$4n^{3/2}$
$3n^{0.5}$	$log\ 5^n$	$3^n + log\ 2023$	2^n	$log_4\ n^n$
4^n	$2n\ log^2\ n$	$2^n + n^2 + 3n$	$\sqrt{(log\ n)}$	$(n+1)!$

From least amount of time to most

$1/n, log(log(n)), x^{.01}, \sqrt{log(n)}, log(5n), 2^{log(n)}, log^2(n), \sqrt{n}, 3x^{.5}, log(5^n), log_4(n^n), 2nlog^2(n), 6nlog(n), 4n^{\frac{3}{2}}, 2^n, 2^n + n^2 + 3n, 3^n + log(2023), 4^n, (n+1)!, 2^{2^x}$.

2. **(20 points)** Find the Asymptotic time complexity for each routine listed in this Python code

    1. O(m)
    2. O(n+m)
    3. O(2n)
    4. O(logn)
    5. O(log(nlog(m)))
    6. O(a/b)

4. 1. **(20 points)** A shelf contains some products sorted according to their width, e.g. |||10, 20|||.

We place more already sorted products with lesser width than the ones that are already present.

e.g. |||10, 20||1, 2, 4, 7|||.
Find the position of a given product (no sorting allowed and find it in O(log n) time).

e.g. |||10, 20||1, 2, 4, 7|||
position of product width 10 is 0
position of product width 7 is 5
to find the product of a certain width, we first look at the first product width in the shelf, if its larger than the width we are looking for, move onto the next shelf, repeat until you find one less than the width, next check the last width, repeating until you find a shelf with the end larger and front smaller and then implement binary search to find wanted width within the shelf.

```
inbetween = False

while inbetween == False:
    if first item > wanted > last item:
        move to next shelf
    if in between those 2 values
        inbetween = True
if wanted > middle
    move middle to middle of higher side and repeat
if wanted < middle
    move middle to middle of lower side and repeat

run until you get the position
```