QUESTION 1: Show that 2^{n+1} is O(2^n).

The Big O notation is expressed using a mathematical formula that uses the letter "O" and a function of "n," which represents the input size.

$2^{n+1} \leq c.2^n$ Where n≥n0

Let's multiply 2^n with 2

$$2^{n+1} = 2.2^n$$

For the constant "c" we write it as

$$2.2^n \leq c.2^n$$

Now, let's divide both sides with 2^n

$$2 \leq c$$

This equation shows us that any value that is 2 or greater than 2 will work. So, let's put c=2

$$2^{n+1} \leq 2.2^n$$

This shows that 2^{n+1} is O(2^n).

QUESTION 2: In each case below, demonstrate whether the given expression is true or false.

a) $5n^3 \in O\left(n^3\right)$

The Big-O notation ignores the constant factors. At both sides there is n^3 so if we multiply it with any constant like 5 in statement it doesn't affect the Big-O classification.

So, this statement is TRUE.

b) $100n^2 \in O\left(n^4\right)$

This statement clearly shows that $n^2$is dominated by n^4. We know that n^4 grows faster than n^2as n increases. Therefore, $100n^2$ grows slower than n^4which means it is in O(n^4).

So, this statement is TRUE.

c) $\log n^2 \in O(\log\ n)$

By using logarithmic properties we can simplify it as: $\log n^2 = 2\log\ n$

In Big-O notation, constants are ignored because we are interested in the growth rate as n becomes large. The difference between O(log n) and O(2log n) is just a constant factor of 2, which becomes insignificant in terms of asymptotic complexity.

Thus, 2 log n is in O(log n).

So, this statement is TRUE.

d) $(n^2 + 7n - 10)^3 \in O(n^6)$

When we expand the highest degree term will be $(n^2)^3 = n^6$ and on right-hand side its O(n^6).

So, this statement is TRUE.

e) $\sqrt{n} \in O\left((\log n)^3\right)$

$\sqrt{n}$ this grows as $n^{\frac{1}{2}}$ and $O\left((\log\ n)^3\right)$ grows slower than $\sqrt{n}$.

We know that logarithmic function grows slower than polynomial functions.

So, this statement is FALSE.

QUESTION 3: In each row of the following table, write whether f = O(g), or f = Ω(g), or both i.e. f = Θ(g)

	F(n)	**g(n)**	**Answer**
1.	100	17	Θ(g)
2.	10^{200}	n-10^{200}	O(g)
3.	$n^2 \log\ 300$	$n\ \log\ n$	Ω(g)
4.	n^{100}	2^n	O(g)
5.	$n\ \log\ n$	n-100	Ω(g)
6.	$\sqrt{n}$	log n	Ω(g)
7.	$n^{1.01}$	n + 100	Ω(g)
8.	$2\ \log\ n$	$\log\left(n^2\right)$	Θ(g)
9.	$\log\left(n^2\right)$	$(\log\ n)^2$	O(g)

QUESTION 4: Order the following functions of n from the smallest to largest complexity. If two have the same complexity put them in one line next to each other.

1. Constant functionality

10 log 300 , 17 , $2 \log_{10} 10^{200}$

These functions are constant functions do not depend on "n". So, they have the smallest complexity.

2. Logarithmic functions

n log n , $n + 7\log(n^2)$ **,** $n^2 \log n$ **,** $n^2 + 7\log(n^2)$

n^2 grows faster than n, the function with n^2 log n will have higher complexity than those with n log n.

3. Polynomial functions

$(n-5)(n) = n^2 - 5n$, $10n^2 + 15n$, $15n^3 + n\log n$, $n^2 + \sqrt{n}$, $n^4 + n^2 + n^2 \log n$, n^4, $3n^2$, $n^2 \log 300$

The complexity is determined by the highest power of n. Functions with higher power have higher complexity.

4. Exponential functions

2^n , $\frac{3n^8}{n^6} 10^{200}$

These grow exponentially with n, which is much faster than any polynomial function. So, they have the highest complexity among all these functions.

QUESTION 5: [Complexity Classes] Order the following functions of n from the smallest to largest complexity and also identify the complexity class for each. If two have the same complexity put them in one line next to each other. Briefly explain next to each line why these functions are in the same complexity class.

1. Constant functionality

10^{200}, $2\log_{10} 10^{200}$,

These functions are constant functions do not depend on "n". So, they have the smallest complexity and belong to O(1) complexity class.

2. Logarithmic functions

$n^2 \log 300$, $n^2 + 7 \log (n^2)$, $n^2 \log\ n$, $n^2 + n^2 \log n$

n^2 grows faster than n, the function with n^2log n will have higher complexity than those with n log n. They belong to $O(n^2 \log n)$ complexity class.

3. Polynomial functions

3^2, $n^2 + \sqrt{n}$, n^4 , $n^4 + n^2 + n^2 \log n$

The complexity is determined by the highest power of n. Functions with higher power have higher complexity.

3^2 is a constant, so it belongs to the O(1) complexity class.

$n^2 + \sqrt{n}$ is dominated by the n^2term, so it belongs to the $O(n^2)$ complexity class.

$n^4$and $n^4 + n^2 + n^2 \log n$are both dominated by the n^4 term, so they belong to the $O(n^4)$ complexity class.

QUESTION 6: Let f(n) and g(n) by asymptotically positive functions. Prove or disprove each of the following conjectures.

1. f(n) = O(g(n)) implies g(n) = O(f(n)).

Disaprove: If f(n)=O(g(n)), this means f(n) grows asymptotically no faster than g(n). However, this does not imply that g(n) grows no faster than f(n).

2. f(n) +g(n) = Θ(min(f(n),g(n))).

Disaprove: If f(n) and g(n) are different in growth rate, the sum is dominated by the larger function.

3. f(n) = O(g(n)) implies lg(f(n)) = O(lg(g(n))), where lg(g(n)) ≥ 1 and f(n) ≥ 1

for all sufficiently large n.

Prove: If f(n)=O(g(n)), then there exists some constant C such that f(n)≤c.g(n) for sufficiently large n. Taking the logarithm on both sides, we get lg(f(n))≤lg(c)+lg(g(n)). Since lg(c) is a constant, this implies lg(f(n))=O(lg(g(n))).

4. f(n) = O(g(n)) implies 2f(n) = O(2g(n)). **Disprove**: Exponential functions grow much faster than their linear or polynomial counterparts. For example, if f(n)=n and g(n)= n^2, then f(n)=O(g(n)), but $2^{f(n)} - 2^n$ and $2^{g(n)} = 2^{n^2}$. Clearly, 2^n grows much more slowly than 2^{n^2}, so $2^{f(n)} \neq O(2^{g(n)})$.

5. f(n) = O((f(n))2).

Prove: For any function f(n), $f(n^2)$ grows faster than or at least as fast as f(n), so $f(n) - O(f(n)^2)$. For example, if $f(n) = n, then\ f(n)^2 = n^2, and\ clearly\ f(n) = O(n^2)$.

6. f(n) = O(g(n)) implies g(n) = Ω(f(n)).

Prove: This is true by definition of Big-O and Big-Ω notations. If f(n)=O(g(n)), this means that g(n) is an upper bound for f(n). By symmetry, this implies that f(n) is a lower bound for g(n) or g(n)=Ω(f(n)).

7. f(n) = Θ(f(n/2)).

Disprove: Not all functions maintain the same asymptotic behavior when their input is halved. For example, f(n)=n is not Θ(f(n/2)), because f(n/2)=n/2, which is not asymptotically the same as n. In general, functions like polynomials or logarithmic functions do not satisfy this property.

8. f(n) +o(f(n)) = Θ(f(n)).

Prove: By the definition of o(f(n)) any function in o(f(n)) grows asymptotically slower than f(n). Therefore, adding a term from o(f(n)) to f(n) will not change the asymptotic behavior of f(n). Hence, f(n)+o(f(n))=Θ(f(n)).

QUESTION 7: Prove or disprove: it is asymptotically faster to square an n-bit integer than to multiply two n-bit integers.

The statement says, "It is asymptotically faster to square an nnn-bit integer than to multiply two nnn-bit integers."

The statement is false.

Multiplying two n-bit numbers: There are several methods to do this, ranging from the basic way we learn in school (which takes $O(n^2)$ steps) to more advanced algorithms like Karatsuba, which takes fewer steps, and even faster methods like the Fast Fourier Transform (FFT), which takes O(nlog$_{f0}$n) steps.

Squaring an n-bit number: Squaring is a special case of multiplication, where you multiply a number by itself. While squaring may save a few steps compared to multiplying two different numbers, it still follows the same basic algorithm. So, even the fastest methods for squaring take the same O(nlog n) steps as multiplication.

Squaring a number isn't faster than multiplication in terms of how the time grows as the numbers get larger. Both squaring and multiplying two numbers have the same growth rate, meaning they take about the same amount of time for very large numbers. In simple words, squaring and multiplying take the same amount of time when the numbers are really large. So the statement is **not true**.