1. Compute

1. $\sum_{i=0}^{n}\sum_{k=0}^{i} 2^k$

$$\begin{aligned}\sum_{i=0}^{n}\sum_{k=0}^{i} 2^k &= \sum_{i=0}^{n}(2^{i+1}-1)\\ &= \sum_{i=0}^{n}2^{i+1} - \sum_{i=0}^{n}1\\ &= 2\cdot\sum_{i=0}^{n}2^{i} - \sum_{i=0}^{n}1\\ &= 2\cdot(2^{n+1}-1)-(n-1)\end{aligned}$$

2. $4^{log_2(5)} + log^2(8^3)$

$$\begin{aligned}4^{log_2(5)} + log^2(8^3) &= 2^{log_2(5^2)} + 3^2\cdot log_2^2(8)\\ &= 25+3^2\cdot 3^2\\ &= 25+3^4\end{aligned}$$

2. Use L'Hopital's rule to determine the limit of:

$$\begin{aligned}lim_{x\to\infty}\frac{x\cdot lnx^5+4}{(3\cdot x+1)^2} &= lim_{x\to\infty}\frac{5\cdot x\cdot ln(x)+4}{9\cdot x^2+6\cdot x+1}\\ &= 0\end{aligned}$$

3. What is the definition of $f(n) = \Omega(g(n))$.

$$f(n)=\Omega(g(n)) \quad\iff\quad \exists c>0, \exists k\geq 0\, s.t\, f(n)\geq c\cdot g(n)\,\forall n\geq k$$

4. What is the growth of the below function

$$f(n) \quad = \quad n\cdot 2^{log_2(n)}+\sqrt{n}+5\cdot n\cdot log^3(n)+log^2(n^n)$$

1. $\Theta(n^2)$
2. $\Theta(n\cdot log^3(n))$

3. $\Theta(\sqrt{n})$
4. $\Theta(n^2 \cdot log^2(n))$
5. Neither!

$$\begin{aligned} f(n) &= n \cdot 2^{log_2(n)} + \sqrt{n} + 5 \cdot n \cdot log^3(n) + log^2(n^n) \\ &= n^2 + \sqrt{n} + 5 \cdot n \cdot log^3(n) + n^2 \cdot log^2(n) \\ &= \Theta(n^2 \cdot log^2(n)) \end{aligned}$$

(4) is the correct answer.

5. What is the growth of the below function

$$f(n) \quad = \quad 6 \cdot log(n) + 4 \cdot log^4 log^3(n) + 3 \cdot log(n^2) + log^2(n)$$

1. $\Theta(log(n))$
2. $\Theta(log(n^2))$
3. $\Theta(log^4 log^3(n))$
4. $\Theta(log^2(n))$
5. Neither!

$$f(n) \quad = \quad \Theta(log^2(n))$$

(4) is the correct answer.

6. Suppose a machine on average takes 10^{-6} seconds to execute a single algorithm. When does the machine finish executing the below code when $n = 1000$?

```
$for(i = 1; \qquad i \leq n; \qquad i++)$
                $count\_sort(a); \quad //a.length = n, max(a) = log_2(n)$
```

$$\begin{aligned} \sum_{i=1}^{n} \Theta(n + log_2(n)) &= \sum_{i=1}^{n} \Theta(n) \\ &= \Theta(n^2) \end{aligned}$$

$(10^2)^3 \cdot 10^{-6} = 1$

7. Sort the below numbers using insertion sort: (1, 7, 2, 0, 4, 5, 3)

1st phase

1, $\boxed{7}$, 2, 0, 4, 5, 3

2nd phase

1 7 $\boxed{2}$ 0 4 5 3

swap

3rd phase

1 2 7 $\boxed{0}$ 4 5 3

4th phase

0 1 2 7 $\boxed{4}$ 5 3

5th phase

0 1 2 4 7 $\boxed{5}$ 3

6th phase

0 1 2 4 5 7 $\boxed{3}$

7th phase

0 1 2 3 4 5 7

8. Prove that $f(n) = 2 \cdot n \cdot log(n^2) - 6 \cdot log^2(n) + \sqrt{n}$ is $O(n \cdot log(n))$, provide the appropriate C and k constants.

$$\begin{aligned} 2 \cdot n \cdot log(n^2) - 6 \cdot log^2(n) + \sqrt{n} &= 4 \cdot n \cdot log(n) - 6 \cdot log^2(n) + \sqrt{n} \\ &\leq 4 \cdot n \cdot log(n) + 6 \cdot n \cdot log(n) + \sqrt{n} \\ &\leq 4 \cdot n \cdot log(n) + 6 \cdot n \cdot log(n) + n \cdot log(n) \\ &= 11 \cdot n \cdot log(n), n \geq 4 \end{aligned}$$

Let $c = 11$ and $k = 4$ then the statement translates to

$$2 \cdot log(n^2) - 6 \cdot log^2(n) + \sqrt{n} \quad \leq \quad c \cdot n \cdot log(n), n \geq k$$

which by the definition of big-O notation, $2 \cdot n \cdot log(n^2) - 6 \cdot log^2(n) + \sqrt{n}$ is $O(n \cdot log(n))$.

9. Compare the growth of $f(n) = 4^{2^{log_2(n)}}$ and $g(n) = 2^{n+log(n)}$

$$\begin{aligned} f(n) &= 4^n \\ g(n) &= n \cdot 2^n \end{aligned}$$

$$\begin{aligned} lim_{n \to \infty} \frac{2^{2 \cdot n}}{n \cdot 2^n} &= lim_{n \to \infty} \frac{2^n}{n} \\ &= \infty \end{aligned}$$

$f(n) = \omega(g(n))$ this implies $f(n) = \Omega(g(n))$. We also know then that $g(n) = O(f(n))$.

10. Prove transitivity of big-O: if $f(n) = O(g(n))$, and $g(n) = O(h(n))$, then $f(n) = O(h(n))$.

Since $f(n) = O(g(n))$ and $g(n) = O(h(n))$ we have the equalities

$$\begin{aligned} f(n) &\leq c_1 \cdot g(n), n \geq k_1 \\ g(n) &\leq c_2 \cdot h(n), n \geq k_2 \end{aligned}$$

From this we obtain

$$f(n) \quad \leq \quad c_1 \cdot c_2 \cdot h(n), n \geq k'$$

where $k' = max(k_1, k_2)$. Let $c = c_1 \cdot c_2$ and $k = k'$ the statement then translate to

$$f(n) \quad \leq \quad c \cdot h(n), n \geq k$$

which by the definition of O notation, $f(n) = O(h(n))$.

11. What is the growth of $n^2 + 2 \cdot n^2 + 3 \cdot n^2 + 4 \cdot n^2 + \ldots + n^5$?

$$\begin{aligned}
\sum_{i=1}^{n^3} i \cdot n^2 &= n^2 \cdot \sum_{i=1}^{n^3} i \\
&= n^2 \cdot \frac{(n^3) \cdot (n^3 + 1)}{2} \\
&= \frac{n^8 + n^5}{2} \\
&= \Theta(n^8)
\end{aligned}$$

12. Prove $\forall k > 0, \epsilon > 0 \implies log^k(n) = o(n^\epsilon)$.

We prove this by induction on k, let ϵ be a real number greater than 0. When $k = 1$ we have

$$\begin{aligned}
lim_{n \to \infty} \frac{log(n)}{n^\epsilon} &= lim_{n \to \infty} \frac{\frac{1}{n \cdot ln(2)}}{\epsilon \cdot n^{\epsilon - 1}} \\
&= lim_{n \to \infty} \frac{1}{\epsilon \cdot ln(2) \cdot n^\epsilon} \\
&= 0
\end{aligned}$$

Assume that the result holds for all integers n less than or equal to some integer k, we show it must also hold for the $k + 1$ integer.

$$\begin{aligned}
lim_{n \to \infty} \frac{log^{k+1}(n)}{n^\epsilon} &= lim_{n \to \infty} \frac{(k+1) \cdot log^k(n) \cdot \frac{1}{n \cdot ln(2)}}{\epsilon \cdot n^{\epsilon - 1}} \\
&= lim_{n \to \infty} \frac{(k+1) \cdot log^k(n)}{\epsilon \cdot ln(2) \cdot n^\epsilon} \\
&= 0
\end{aligned}$$

which is what we wanted to show. Hence by the principle of mathematical induction the result holds for all $n > 0$ and $\epsilon > 0$.

13. Use to the definition of big-O to prove that $f(n) + g(n) = O(max(f(n), g(n)))$.

$$f(n) + g(n) \leq 2 \cdot max(f(n), g(n)), n \geq 1$$

which by the definition of big-O notation, $f(n) + g(n) = O(max(f(n), g(n)))$.

14. Prove or disprove $(n!)^{log(n)} = \omega(n^{2^{log(n)}})$

$$\begin{aligned}
log(lim_{n\to\infty}\frac{n!^{log(n)}}{n^n}) &= lim_{n\to\infty}log(n)\cdot log(n!) - n\cdot log(n) \\
&= lim_{n\to\infty}\Theta(n\cdot log_2^2(n)) - n\cdot log_2(n) \\
&= \infty \\
2^{\infty} &= \infty
\end{aligned}$$

which by the definition of little-ω notation, $n!^{log(n)} = \omega(n^{2^{log(n)}})$.

15. Given a sorted array of n distinct number where the range of the numbers are between 0 and m and $m > n$. Find the smallest number.

```cpp
#include <iostream>
#include <vector>

using namespace std;

// T(n) = T(n / 2) + O(1)
// T(n) = O(log_2(n))
int binary_search(const vector<int>& A, int start, int end){
    if(start > end) return end + 1;

    if(A[start] != start) return start;

    int mid = (start + end) / 2;

    if(A[mid] == mid) return binary_search(A, mid + 1, end);

    return binary_search(A, start, mid - 1);
}

int main() {
    vector<int> A1 = {0, 1, 3, 6, 8, 9};
    vector<int> A2 = {0, 2, 5, 7, 11};
    cout << binary_search(A1, 0, A1.size() - 1) << endl;
    cout << binary_search(A2, 0, A2.size() - 1) << endl;
}
```