**1.** Compute

1. $\sum_{i=-10}^{n}(\frac{1}{2})^i + \sum_{i=200}^{n^2}(3)^i$

$$
\begin{aligned}
\sum_{i=-10}^{n}(\frac{1}{2})^i + \sum_{i=200}^{n^2}(3)^i &= \sum_{i=-10}^{-1}(\frac{1}{2})^i + \sum_{i=0}^{n}(\frac{1}{2})^i + \sum_{i=0}^{n^2}(3^i) - \sum_{i=0}^{199}(3^i) \\
&= \sum_{i=-10}^{0}(\frac{1}{2})^i - \sum_{i=0}^{0}(\frac{1}{2})^i + \sum_{i=0}^{n^2}(3^i) - \sum_{i=0}^{199}(3^i) \\
&= -\sum_{i=0}^{10}(\frac{1}{2})^i - 1 + \frac{3^{n^2+1}-1}{3-1} - \frac{3^{200}-1}{3-1} \\
&= -\frac{(\frac{1}{2})^{11}-1}{\frac{1}{2}-1} - 1 + \frac{3^{n^2+1}-1}{3-1} - \frac{3^{200}-1}{3-1}
\end{aligned}
$$

2. $7^{log_2 log_2 4} + log_3 log_2^2 8$

$$
\begin{aligned}
7^{log_2 log_2 4} + log_3 log_2^2 8 &= 7^{log_2 2} + log_3 9 \\
&= 7 + 2 \\
&= 9
\end{aligned}
$$

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

$$lim_{x\to\infty}\frac{xlnx^2+3x}{\sqrt{4x^2-1}}$$

$$
\begin{aligned}
lim_{x\to\infty}\frac{xlnx^2+3x}{\sqrt{4x^2-1}} &= lim_{x\to\infty}\frac{2\cdot x\cdot lnx + 3\cdot x}{\sqrt{4\cdot x^2-1}} \\
&= lim_{x\to\infty}\frac{2\cdot lnx+3}{\frac{1}{2}(4\cdot x^2-1)^{-\frac{1}{2}}}\cdot 8\cdot x \\
&= \infty
\end{aligned}
$$

**3.** What is the growth of the below function

$$f(n) \quad = \quad 8^{log_2 n} + \sqrt{n^6 log_2 n} + nlog_2^8 n + log_2 n^{2^{log_2 n}}$$

1. $\Theta(n^3)$

2. $\Theta(n^3 log_2 n)$

3. $\Theta(n^3 \sqrt{log_2 n})$

4. $\Theta(n log_2 n)$

5. Neither!

$$\begin{aligned} f(n) &= 8^{log_2 n} + \sqrt{n^6 log_2 n} + n \cdot log_2^8 n + log_2 n^{2^{log_2 n}} \\ &= 2^{log_2 n^3} + n^3 \cdot \sqrt{log_2 n} + n \cdot log_2^8 n + n \cdot log_2 n \\ &= n^3 + n^3 \cdot \sqrt{log_2 n} + n \cdot log_2^8 n + n \cdot log_2 n \end{aligned}$$

Which is $\Theta(n^3 \sqrt{log_2 n})$.

**4.** What is the growth of the below function.

1. $\Theta(log_2 n)$

2. $\Theta(log_2 log_2^6 n)$

3. $\Theta(log_2 n^{10})$

4. $\Theta(log_2^2 n)$

5. Neither!

$$\begin{aligned} f(n) &= 2^{log_2 log_2 n} + 3 log_2 log_2^6 n + 5 log_2^2 n + log_2 n^{10} \\ &= log_2 n + 3 \cdot log_2(log_2^6 n) + 5 \cdot log_2^2 n + 10 \cdot log_2 n \end{aligned}$$

which is $\Theta(log_2^2 n)$.

**5.** Suppose a machine on average takes $10^{-6}$ seconds to execute a single algorithm how long does it take for the machine to finish executing the code below when $n = 100$?

$$\begin{array}{ccc} for(i = 0; & i < n^2; & i++) \\ for(k = 0; & k < i; & k++) \\ & selection\_sort(a); & //a.length == n \end{array}$$

$\sum_{i=1}^{n^2} \sum_{k=1}^{i} \Theta(n^2) = \sum_{i=1}^{n^2} i\Theta(n^2) = \Theta(n^6)$. $(10^2)^6 \cdot 10^{-6} = 10^6$ seconds.

**6.** Assume you want to write a code to calculate the multiplication of two numbers. Provide the running time for your algorihm, assuming the inputs are two n-digits numbers.

Using the grade school multiplcation algorithm we can achieve a run-time of $\Theta(n^2)$.

**7.** Sort the below numbers using radix sort:

(2) 14, 10, 32, 50, 1, 54, 12, 45

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 50 | | | | 54 | | | | | |
| 10 | 1 | 32 | | 14 | 45 | | | | |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

1st 10 50 1 32 14 54 45

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | 14 | | | | 54 | | | | |
| 1 | 10 | | 32 | 45 | 50 | | | | |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

2nd pass 1 10 14 32 45 50 54

**8.** Prove that $f(n) = log^2(n) - 6 \cdot log(log(n)) + 4 \cdot log(n)$

$$\begin{aligned} log^2(n) - 6 \cdot log(log(n)) + 4 \cdot log(n) &\leq log^3(n) + 6 \cdot log^3(n) + 4 \cdot log^3(n) \\ &= 11 \cdot log^3(n) \end{aligned}$$

Hence we conclude $f(n) = O(log^3(n))$.

**9.** Prove that if $f(n) = \Theta(h(n)), g(n) = \Theta(k(n))$ then $f(n) \cdot g(n) = \Theta(h(n) \cdot k(n))$.

$$\begin{aligned} f(n) &\leq c_1 \cdot h(n) \\ f(n) \cdot g(n) &\leq c_1 \cdot g(n) \cdot h(n) \\ &\leq c_1 \cdot c_2 \cdot h(n) \cdot k(n) \end{aligned}$$

hence $f(n) \cdot g(n) = O(h(n) \cdot k(n))$.

$$\begin{aligned} c_1 \cdot h(n) &\leq f(n) \\ c_1 \cdot g(n) \cdot h(n) &\leq f(n) \cdot g(n) \\ c_1 \cdot c_2 \cdot k(n) \cdot h(n) &\leq f(n) \cdot g(n) \end{aligned}$$

hence $f(n) \cdot g(n) = \Omega(h(n) \cdot k(n))$. We conclude $f(n) \cdot g(n) = \Theta(h(n) \cdot k(n))$.

**10.** Compare the growth of $f(n) = \sqrt{n} \cdot log^2(n), g(n) = n^{2+sin(n)}$

(10)

$$\frac{\sqrt{n}\,\log^2(n)}{n^2 \cdot n^{\sin(n)}}$$

Case 1: $\sin(n) = -1$

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$$

Case 2: $\sin(n) = 0$

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$$

Case 3 $\sin(n) = 1$

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$$

Case 4 $0 \leq \sin(n) < 1$

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0 \quad f(n) = o(g(n))$$

$$g(n) = \frac{c}{c\,g(n)}$$

Case 5: $-1 < \sin(n) < 0$

$$\lim \frac{f(n)}{g(n)} = 0$$

**11.** What is the growth of $\sum_{i=1}^{n} i \cdot ln(i)$

$$\begin{aligned}
\sum_{i=1}^{n} i \cdot ln(i) &= \Theta(\int_1^n x \cdot ln(x) \cdot dx) \\
&= \Theta(n^2 \cdot ln(n))
\end{aligned}$$

**12.** Prove that if $f(n)$ is monotonically decreasing, then

$$\sum_{i=1}^{n} f(i) = \Omega(\int_1^n f(x) \cdot dx)$$

$\int_1^n f(x) \cdot dx \leq \sum_{i=1}^{n-1} f(i) \leq \sum_{i=1}^{n} f(i)$ hence $\sum_{i=1}^{n} f(i) = \Omega(\int_1^n f(x) \cdot dx)$.

**13.** Prove or disprove: if $f(n) = O(g(n))$ and $f(n) \geq 1$ and $log(g(n)) \geq 1$ for sufficiently large $n$ then $log(f(n)) = O(log(n))$.

$$\begin{aligned}
f(n) &\leq c \cdot g(n) \\
log_2(f(n)) &\leq log_2(c) + log_2(g(n)) \\
log_2(f(n)) &\leq log_2(c) \cdot log_2(g(n)) + log_2(g(n)) \\
&= log_2(g(n)) \cdot (log_2(c) + 1)
\end{aligned}$$

hence $log_2(f(n)) = O(log_2(n))$.

**14.** Prove or disprove $log_2(n)^{2 \cdot log^3(n)} = \omega((n!)^2)$

$$\begin{aligned}
lim_{n \to \infty} 2 \cdot log^3(n) \cdot log_2(log_2(n)) - 2 \cdot log(n!) &= 2 \cdot log^3(n) \cdot log_2(log_2(n)) - 2 \cdot \Theta(n \cdot log_2(n)) \\
&= \infty \\
2^{\infty} &= \infty
\end{aligned}$$

Hence $log_2(n)^{2 \cdot log^3(n)} = \omega((n!)^2)$.

**15.** Given a sorted array with $n$ integers, provide an algorithm with the running time of $O(log_2(n))$ that checks if there is n $i$ for which $a[i] = i$.

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

using namespace std;

// T(n) = T(n / 2) + O(1)
```

```cpp
// T(n) = O(log_2(n))
int binary_search(const vector<int>& A){
    int lo = 0, hi = A.size() - 1;
    while(lo <= hi){
        int mid = (lo + hi) / 2;
        if(A[mid] < mid){
            lo = mid+ 1;
        } else if(A[mid] > mid){
            hi = mid - 1;
        } else {
            return mid;
        }
    }
    return -1;
}

int main() {
    vector<int> A = {0, 1, 2, 5, 10, 21};
    cout << binary_search(A) << endl;
}
```