CMSC 341 Data Structure
Asymptotic Analysis Review

1. What is the purpose of asymptotic analysis?

2. Define “Big-Oh” using a formal, mathematical definition.

3. Let $T_1(x) = O(f(x))$ and $T_2(x) = O(g(x))$.
   Prove $T_1(x) + T_2(x) = O(\max(f(x), g(x)))$

4. Let $T(x) = O(cf(x))$. Prove $T(x) = O(f(x))$.

5. Let $T_1(x) = O(f(x))$ and $T_2(x) = O(g(x))$.
   Prove $T_1(x) * T_2(x) = O(f(x) * g(x))$.

6. Prove $2^{n+1} = O(2^n)$.

7. Prove that if $T(n)$ is a polynomial of degree $x$, then $T(n) = O(n^x)$.

8. Number these functions in ascending (slowest growing to fastest growing) Big-Oh order

<table>
<thead>
<tr>
<th>Number</th>
<th>Big-Oh</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$O(n^3)$</td>
</tr>
<tr>
<td></td>
<td>$O(n^2 \lg n)$</td>
</tr>
<tr>
<td></td>
<td>$O(1)$</td>
</tr>
<tr>
<td></td>
<td>$O(\lg n)$</td>
</tr>
<tr>
<td></td>
<td>$O(n^{1.01})$</td>
</tr>
<tr>
<td></td>
<td>$O(n^{2.01})$</td>
</tr>
<tr>
<td></td>
<td>$O(2^n)$</td>
</tr>
<tr>
<td></td>
<td>$O(\lg n)$</td>
</tr>
<tr>
<td></td>
<td>$O(n)$</td>
</tr>
<tr>
<td></td>
<td>$O(n \lg n)$</td>
</tr>
<tr>
<td></td>
<td>$O(n \lg^2 n)$</td>
</tr>
</tbody>
</table>
9. Determine, for the typical algorithms that you use to perform calculations by hand, the running time to
   (a) Add two N-digit numbers
   (b) Multiply two N-digit numbers

10. What is the asymptotic performance of each of the following?
    Select among
    (A) $O(n)$  (B) $O(n^2)$  (C) $O(n \log n)$  (D) $O(n^3)$
    (E) $O(\log n)$  (F) $O(1)$  (G) $O(n!)$ (H) none of these
    (a) ______ Squaring each element of an $N \times N$ matrix.
    (b) ______ Finding the smallest value in a sorted array of $N$ integers
    (c) ______ Finding a value in a sorted array using binary search
    (d) ______ Pushing $N$ elements onto a stack, then popping them and printing them
    (e) ______ Finding the largest 3 values in an unsorted array

11. What is the asymptotic performance of the following C++ code fragment?
    Justify your answer.
    ```
    for (int i = 0; i < N; i++)
    {
        for (int j = 10; j >= 0; j--)
        {
            int count = 1;
            while (count < N)
            {
                count *= 2;
            }
        }
    }
    ```