**CS 213**

**Homework Assignment 1**

**Given:** February 03, 2015

**Due:** February 10, 2015

---

**Note:** The homework is due on Thursday, February 10 at the beginning of the class.

Please write concise and clear solutions; you will get only a partial credit for correct solutions that are either unnecessarily long or not clear.

You are allowed to discuss ideas for solving homework problems in groups of up to 3 people but **you must write your solutions independently**. Also, you must write on your homework the names of the people with whom you discussed.

Finally, you are not allowed to use any material outside of the class notes and the textbook. Any violation of this policy may seriously affect your grade in the class.

---

In this homework, \( \log n \) refers to \( \log_2 n \) and \( \lg n \) also refers to \( \log_2 n \).

\[ 15 \] **1.** (a) For each pair of expressions \((A,B)\) in the table below, indicate whether \( A \) is \( O, o, \Omega, \omega, \text{or } \Theta \) of \( B \). Assume that \( k \geq 1, \epsilon > 0, \text{and } c > 1 \) are constants. Your answer should be in the form of the table with “yes” or “no” written in each box. No justification is required.

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>O</th>
<th>o</th>
<th>Ω</th>
<th>ω</th>
<th>Θ</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \lg^k n )</td>
<td>( n^\epsilon )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( n^k )</td>
<td></td>
<td>( \epsilon n )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( 2^n )</td>
<td></td>
<td>( 2^{n/2} )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( n! \sqrt{\lg n} )</td>
<td>( c^{\lg n} )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( \lg(n!) )</td>
<td>( \lg(n^n) )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(b) Rank the following functions by order of their growth; i.e., find an arrangement \( g_1, g_2, \ldots \) of the functions satisfying \( g_1 = \Omega(g_2), g_2 = \Omega(g_3), \ldots \). No justification is required.

\[
\begin{align*}
(\sqrt{2})^{\lg n} & \quad n^2 & \quad n! & \quad (\lg n)! \\
(3/2)^n & \quad n^3 & \quad \lg^2 n & \quad \lg(n!) \\
2^{2^n} & \quad n^{1/\lg n} & \quad \ln \ln n & \quad n \cdot 2^n \\
n^{\lg \lg n} & \quad \ln n & \quad 1 & \quad 2^{\lg n} \\
(\lg n)^{\lg n} & \quad e^n & \quad 4^{\lg n} & \quad (n + 1)! \\
\sqrt{\lg n} & \quad 2^{\sqrt{\lg n}} & \quad n & \quad 2^n \\
n \lg n & \quad 2^{2^{n+1}} & \quad 2^{100^{100}} & \quad \text{ } \\
\end{align*}
\]
2. Prove or disprove the following. In case of a proof, use the definitions of $O$, $\Omega$, and $\Theta$ and give values of the constants in the definitions for which the conditions in the definition hold.

   (i) $5n\sqrt{n} = O\left(\frac{1}{2}n^2 - 10\right)$
   (ii) $2^{5\lg n + \lg \lg n} \lg(n^5) = O(4^{3\lg n})$

3. Let $f(n)$ and $g(n)$ be two functions from the set of positive integers to the set of positive integers. Prove or disprove the following.

   (i) There exist functions $f(n)$ and $g(n)$ such that neither $f(n)$ is $O(g(n))$ nor $g(n)$ is $O(f(n))$.
   (ii) If $f(n) = O(s(n))$ and $g(n) = O(r(n))$ then $f(n)/g(n) = O(s(n)/r(n))$.

4. Consider the following code fragment.

   ```java
   for i = 1 to n + 100 do
      for j = 1 to i * n do
         sum = sum + j
      for k = 1 to 3n do
         c[k] = c[k] + sum
   ```

   a. For the above code fragment give a bound of the form $O(f(n))$ on its running time on an input of size $n$. Justify your answer.

   b. For this same function $f$, show that the running time of the algorithm on an input of size $n$ is also $\Omega(f(n))$. Justify your answer. (This shows an asymptotically tight bound of $\Theta(f(n))$ on the running time.)

5. Consider the following code fragment.

   ```java
   sum = 0;
   for (int i=1; i <= n; i++)
      for (int j=1; j <= i; j*=2)
         for (int k=1; k <= j; k*=3)
            sum++;
   ```

   a. For the above code fragment give a bound of the form $O(f(n))$ on its running time on an input of size $n$. Justify your answer.

   b. For this same function $f$, show that the running time of the algorithm on an input of size $n$ is also $\Omega(f(n))$. Justify your answer. (This shows an asymptotically tight bound of $\Theta(f(n))$ on the running time.)
6. Consider the following basic problem. You’re given an array $A$ consisting of $n$ integers $A[1], A[2], \ldots, A[n]$. You’d like to output a two-dimensional $n$-by-$n$ array $B$ in which $B[i,j]$ (for $i < j$) contains the sum of array entries $A[i]$ through $A[j]$ – that is, the sum $A[i] + A[i+1] + \cdots + A[j]$. (The value of the array entry $B[i,j]$ is left unspecified whenever $i \geq j$, so it doesn’t matter what is the output of these values.)

Here is a simple algorithm to solve this problem.

```plaintext
for (i=1; i <= n; i++)
    for (j=i+1; j <= n; j++)
        Add up array entries $A[i]$ through $A[j]$;
        Store the result in $B[i,j]$
```

a. For the above code fragment give a bound of the form $O(f(n))$ on its running time on an input of size $n$. Justify your answer.

b. For this same function $f$, show that the running time of the algorithm on an input of size $n$ is also $\Omega(f(n))$. (This shows an asymptotically tight bound of $\Theta(f(n))$ on the running time.)

c. Although the algorithm you analyzed in parts (a) and (b) is the most natural way to solve the problem – after all, it just iterates through the relevant entries of the array $B$, filling in a value for each – it contains some highly unnecessary sources of inefficiency. Give a different algorithm to solve this problem, with an asymptotically better running time. In other words, you should design an algorithm with running time $O(g(n))$, where $\lim_{n \to \infty} g(n)/f(n) = 0$. 