1. Running time analysis, 20 points total
Rank the following functions in order of increasing asymptotic growth rate, starting with 1 (i.e. 1 will be the slowest growing function). If two functions have the same asymptotic growth rate give them the same number.

<table>
<thead>
<tr>
<th>$n/\log n$</th>
<th>$2 + \sin^n n$</th>
<th>$\log n + 999$</th>
<th>$n\log n$</th>
<th>$n^{1/2} \log^{1000} n$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$n^{1/\log n}$</td>
<td>$\log(n!)$</td>
<td>$3^n$</td>
<td>$n^{2^n}$</td>
<td>$2^n$</td>
</tr>
</tbody>
</table>

2. Asymptotic bounds, 10 points total
a. If you wanted to prove that $n^2 \log(n^{2.5}) \in O(n^2 \log n)$ what values of $c$ and $n_0$ would work?

b. If you wanted to prove that $n^2 \log(n^{2.5}) \in \Omega(n^2 \log n)$ what values of $c$ and $n_0$ would work?

3. More asymptotics, 10 points total
Do functions $f(n)$ and $g(n)$ exist that satisfy the following relationships? If so, find a pair of such functions.

a. $f(n) \notin \Theta(g(n))$ and $g(n) \in \Omega(f(n))$

b. $f(n) \in O(g(n))$ and $\log(f(n)) \in O(\log(g(n)))$

c. $f(n) \in O(g(n))$ and $(f(n))^2 \in O((g(n))^2)$

d. $f(n) \in O(g(n))$ and $2^{f(n)} \in O(2^{g(n)})$

4. Array manipulation, 30 points total
Consider an algorithm that takes an array of integers of length $n$, $A[1, .., n]$ and outputs a two-dimensional array of integers $B$. Each $B[i, j]$ element consists of the sum of array elements $A[i]$ through $A[j]$. If $i \geq j$ then the value of array element $B[i, j]$ is left unspecified. A pseudocode of the algorithm is as follows:
1. \textbf{for} \ i \gets 1 \ \text{to} \ n
2. \ \textbf{for} \ j \gets (i+1) \ \text{to} \ n
3. \quad \text{Add up array entries } A[i] \ \text{through} \ A[j]
4. \quad \text{Store the result in} \ B[i,j]

a. What are the upper and lower bounds, in big-O and big-$\Omega$ notation, on the running time of this algorithm?

b. Can you come up with a more efficient algorithm for this problem? You do not need pseudocode, just explain the algorithm in a clear manner.

5. \textbf{Reduction, 30 points total}
In computer science we can often solve an instance of one problem by reducing it to a second problem. The solution to the second problem, if the reduction is done right, can give us a solution to the original problem. Using this idea, show how interval scheduling and bipartite matching can be reduced to independent set such that the solution of independent set gives the solution to the original problems.