Exercise 1:  
Consider the following functions, and rearrange them in ascending order of growth rate. That is, if function $g(n)$ comes after function $f(n)$ in your list, then it should be the case that $f(n) = O(g(n))$. 

$$
\begin{align*}
  f_1(n) &= n^{\log n} \\
  f_2(n) &= 7n - 2 \\
  f_3(n) &= 100\sqrt{n} + 2n^{1/4} - 100 \\
  f_4(n) &= 2^{\log^2 n} \\
  f_5(n) &= 2^{2\log n} + n^{1.1} \\
  f_6(n) &= 3^n \\
  f_7(n) &= n \log n
\end{align*}
$$

Exercise 2:  
Assume you have functions $f$ and $g$ such that $f(n) = O(g(n))$. For each of the following statement, decide whether you think it is true of false and give a proof or counterexample. You may assume that all functions mentioned are monotone non-decreasing and non-negative.

(a) $\log_2 f(n) = O(\log_2 g(n))$

(b) $2^{f(n)} = O(2^{g(n)})$

(c) $f(n)^2 = O(g(n)^2)$
Exercise 3: (20 Points)
You are 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 array entry $B[i, j]$ with $i \geq j$ is left unspecified, so it doesn’t matter what to output for these values.) Here is a simple algorithm to solve this problem.

\[
\text{for } i = 1 \text{ to } n \text{ do do}
\]

\[
\text{for } j = i + 1 \text{ to } n \text{ do do}
\]

\[
\text{Add up array entries } A[i] \text{ through } A[j];
\]

\[
\text{Store the result in } B[i, j];
\]

\[
\text{end}
\]

\[
\text{end}
\]

(a) Find a function $f$ such that the running time of this algorithm is $\Theta(f(n))$.

(b) The above algorithm 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 $g(n) = o(f(n))$.

Exercise 4: (20 Points)
A binary tree is a rooted tree in which each vertex has at most two children. Show by induction that in any binary tree the number of vertices with two children is exactly one less than the number of leaves (a leaf is a vertex without any children).

Exercise 5: (20 Points)
Let $G$ be a connected undirected graph, and let $T$ be a depth-first search tree of $G$ rooted at some node $v$. Prove that if $T$ is also a breadth-first search tree of $G$ rooted at $v$, then $G = T$