1. Big-O Review

For each of the following, decide whether the statement is true or false, and justify your answer. Recall that $f(x) = O(g(x))$ if there are constants $C$ and $k$ such that $|f(x)| \leq C|g(x)|$ whenever $x > k$. You may assume $n$ is a positive integer.

(a) [3 points] $3n \log_{10}(n) + 2n + 100 = O(n \log_2(n))$

(b) [3 points] $2^{2n} = O(2^n)$

(c) [3 points] $\log(n!) = O(n \log n)$ (Hint: what is the relationship between $n!$ and $n^n$?)

2. More Big-O Review [6 points]

Consider the following table, which contains the runtime in milliseconds for three different algorithms with five different input sizes (10, 20, 50, 1000, 2000). For each algorithm, give a big-O bound on the runtime.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>10</th>
<th>20</th>
<th>50</th>
<th>1000</th>
<th>2000</th>
</tr>
</thead>
<tbody>
<tr>
<td>Algorithm 1</td>
<td>50</td>
<td>110</td>
<td>900</td>
<td>1990</td>
<td>3000</td>
</tr>
<tr>
<td>Algorithm 2</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>Algorithm 3</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
</tbody>
</table>

3. Real-World Multiplication [3 pts] I pulled the following text from python source code:

```c
/* For long multiplication, use the O(N**2) school algorithm unless
 * both operands contain more than KARATSUBA_CUTOFF digits (this
 * being an internal Python long digit, in base PyLong_BASE).
 */
#define KARATSUBA_CUTOFF 70 (1)
```

Explain what choice is being made here, and why.

4. Inductive Proofs Review
(a) **[11 points]** Use induction to prove that for all \( n \geq 0 \):

\[
1 + r + r^2 + r^3 + \ldots + r^n = \frac{r^{n+1} - 1}{r - 1}
\]  

where \( r \) is any number not equal to 1. Make sure your inductive proof clearly denotes the base case and the inductive step.

(b) **[3 points]** Where in your proof did you use the fact that \( r \neq 1 \)?

(c) **[2 points]** What does the sum evaluate to when \( r = 1 \)?

5. Let \textit{SelfReference} be an algorithm that takes as input a sorted (in increasing order) array \( A \) of \( n \) distinct integers, and returns an index \( i \) such that \( A[i] = i \), or returns 0 otherwise. (Assume the indices of \( A \) start at 1 and go to \( n \).)

   (a) **[9 points]** Write pseudocode for a recursive version of \textit{SelfReference} that is as fast as possible. (Your algorithm can take additional inputs if you find it helpful.)

   (b) **[11 points]** Prove your algorithm is correct.

   (c) **[3 points]** What is the asymptotic runtime of your algorithm? Take 3 bonus points if your algorithm is as fast or faster than mine.

6. Approximately how long did you spend on this assignment (round to the nearest hour)?