1. **Time analysis**

What is the running time of the following algorithm (using big-O notation)? Show your analysis.

```
1: result ← 1
2: for i = 1 to n do
3:     for j = 1 to i do
4:         result ← result + 1
5:     end for
6: end for
```

2. **Asymptotic Notations**

1. Show $8n^3\log n + 14n^2 = \Theta(n^3\log n)$.

2. If $f(n) = O(g(n))$, can we conclude $2f(n) = O(2g(n))$? Justify your answer.

3. Prove for any integer constant $a$ and real constant $b$, $(n + b)^a = \Theta(n^a)$. In fact, this holds even when $a$ is a real constant, but you do not have to prove it.

3. **Search in O(\log n) time**

Given a sorted array of distinct integers $A[1..n]$, you want to find out whether there is an index $i$ for which $A[i] = i$. Give an algorithm that runs in time $O(\log n)$. Note: you should explain why the algorithm is correct and then analyze the running time.

4. **Factorial**

In this problem, we suppose the running time is proportional to the number of bit additions involved. Recall that in class we discussed the integer multiplication problem: the running time for multiplying two integers of $n$ bits each takes $O(n^2)$ time. It is not hard to generalize this: for two integers with $n_1$ and $n_2$ bit respectively, we can compute their product in $O(n_1n_2)$ time. You can use this fact without a proof.

Now, consider the problem of computing $N! = 1 \times 2 \times 3 \times \ldots \times N$, where $N$ is an $n$-bit integer. You perhaps programmed the following recursion when you took your first programming class.

```
function Factorial(n)
    if n = 1 then return 1 else return n*Factorial(n-1);
```

1. How many multiplications are there in computing $\text{Factorial}(N)$?

2. Show the number of bits you need to store $N!$ is $\Theta(N\log N)$. If you are unfamiliar with factorial, review p. 54-55.

3. What is the running time of the algorithm in computing $N!$? Show your analysis.