CS109A Problem Set #3

Due 4:30 PM, February 1, 1995

1. (25 points) For each of the following, determine whether $f(n)$ is $O(g(n))$. Prove your answer.

   (a) $f(n) = n^2$; $g(n) = 1000n$.

   (b) $f(n) = 6^n$; $g(n) = 2^n$.

   (c) $f(n) = 3n^4 + n^3 + 4n^2 + n + 6$; $g(n) = n^4$.

   (d) $f(n) = \text{if } (n \text{ is a power of } 2) \text{ then } n^3 \text{ else } n$; $g(n) = n^2$.

   (e) $f(n) = \log_2 n$; $g(n) = \sqrt{n}$.

2. (25 points) The textbook claims that any exponential grows faster than any polynomial (p. 102, FCS). We now prove that claim. This distinction between exponential and polynomial time is fundamental to more sophisticated concepts in computer science regarding whether or not a problem can be solved efficiently. For all a and for all $b > 1$:

   (a) Prove that if $n > 1/(\sqrt[a]{b} - 1)$, then $(n+1)^a/n^a < b$. This shows that for large enough n, the function n^a grows slower than b^n does.

   (b) Prove that if $n > (a/\ln b)^2$, then $n^a < b^n$. You may use the fact that $\sqrt{n} > \ln n$.

   (c) Choose $n_0 > \max(1/(\sqrt[a]{b} - 1), (a/\ln b)^2$, and prove that n^a is $O(b^n)$.

   (d) Now, prove that b^n is not $O(n^a)$. Hint: choosing $n_1 > \max(n_0, c, ((a+1)/\ln b)^2)$ works.

3. (25 points) The following C function computes and prints all prime numbers less than or equal to n. Draw the structure tree for the function, and give a big-oh upper bound on running time for each node of the structure tree.

```
        void print_primes(int n)
        {
            int i, candidate, is_prime;
(1)         for (candidate=2; candidate<n; candidate++) {
(2)             is_prime=TRUE;
(3)             for (i=2; i*i<=candidate; i++)
(4)                 if (candidate%i==0)
(5)                     is_prime=FALSE;
(6)             if (is_prime)
(7)                 printf("%d\n",candidate);
            }
        }
```

4. (25 points) *Fibonacci numbers* are the sequence formed by starting with two 1s and generating each successive term by adding the previous two terms, e.g., 1, 1, $2(= 1+1)$, $3(= 2+1)$, $5(= 3+2)$, 8, 13, 21, 34 .... They are a favorite example in computer science classes, but also show up surprisingly often in real-life computer science and even in art and nature. The following ML function computes the nth Fibonacci number:

```
fun fib(n) = if n<=2 then 1 else fib(n-1)+fib(n-2);
```

   (a) Find a recurrence relation for $T_{\text{fib}}(n)$, the running time required by the above function.

   (b) Prove that $T_{\text{fib}}(n)$ is $O(2^n)$.