CSC 238F 1999, Assignment 2, St. George Campus
Due: Oct. 20, 10:10 AM (beginning of tutorial)

On the cover page of your assignment, you must write **and sign** *the following statement: "I have read and understood the policy on collaboration in homework stated in the webpage of the course." Without such a signed statement your homework will not be marked.*

---

1. Consider the following program, where all variables hold integers. The goal is to divide a by b, using (essentially) the "long division" algorithm. In order to understand (and efficiently implement) this algorithm, it is convenient to think of the integers as being represented in binary (base 2) notation. (Note that we assume, in the Precondition below, that we have already calculated into the variable x a sufficiently large "shifted" version of b, and that y has been assigned the appropriate power of 2 corresponding to how much this shift is.) [15]

```
q := 0;    r := a;
while r ≥ b
    if r ≥ x then
        r := r − x;
        q := q + y;
    end if;
    x := x div 2;    y := y div 2;
end while
```

Prove that the program is correct with respect to the following Precondition/Postcondtion pair.

Precondition:
$a, b \in \mathbb{N}$ and $b > 0$.
For some $k \in \mathbb{N}$, $x = b \cdot 2^k$ and $y = 2^k$.
$x > a$.

Postcondition:
$q = a \text{ div } b$.

HINT: Your loop invariant should include the requirement that x be large enough.

2. In the following program, A is an integer array and all other variables are integer variables. For convenience, we assume that $A(x)$ has a value for every integer x. We wish to find the last position between F and L that contains the minimum value between F and L. (Note that A may contain several occurrences of the same integer in different positions.)

```
MIN(A, F, L)
    if F = L then
        return F
    else
        M := (F + L) div 2;
        x := MIN(A, F, M);
        y := MIN(A, M + 1, L);
        if A(y) ≤ A(x) then
            return y
        else
            return x
        end if
    end if
end MIN
```

(a) Prove *carefully* that the program is correct with respect to the following Precondition/Postcondtion pair. [10]

Precondition:
$F \leq L$

Postcondition: A (integer) value u is returned such that
$F \leq u \leq L$, and
$A(u)$ is the smallest value in $\{A(v) | F \leq v \leq L\}$, and
$A(u) < A(v)$ for every v such that $u < v \leq L$.

(b) Let $T(n)$ be the running time of the program when $n = L - F + 1$. Give a recurrence for T. [5]

(c) Use the general theorem about divide-and-conquer recurrences to give (using O notation) an upper bound on T. [5]

3. Consider the following recurrence defining a function from $\mathbb{N}$ to $\mathbb{N}$.

$T(0) = 0,$
$T(1) = 1,$
$T(n) = 4T(\lfloor n/3 \rfloor) + n + 3$, for all $n > 1$.

We know from the general theorem about solving divide-and-conquer recurrences that $T(n) \in O(n^{\log_3 4})$. We are interested in finding the smallest constant c (if there is one) such that except for finitely many values of $n \in \mathbb{N}$, $T(n) \leq c \cdot n^{\log_3 4}$.

(a) Give a constant c and integer n_0, and use induction to prove that for all $n \geq n_0$, $T(n) \leq c \cdot n^{\log_3 4}$. [5]

HINT: There are many ways to approach this. One way is to prove that for all $n \geq 1$, $T(n) \leq 8 \cdot n^{\log_3 4} - 3 \cdot n - 4$. You may have to use the inequalities:
$\lfloor \frac{n}{3} \rfloor \leq \frac{n}{3}$, and $\lfloor \frac{n}{3} \rfloor \geq \frac{n-2}{3}$.

(b) It is easiest to prove an upper bound for $T(n)$ for the case when n is a power of 3. Prove that for all n of the form 3^k where $k \in \mathbb{N}$,
$T(n) \leq 5 \cdot n^{\log_3 4}$. [5]

HINT: Prove, for some appropriate d and e, that for every n of the form 3^k,
$T(n) \leq 5 \cdot n^{\log_3 4} - d \cdot n - e$.

(c) It turns out that the constant "5" is not good enough for all n. Let $c = \frac{12}{2^{\log_3 4}}$; c is about 5.004. Prove that for every $c' < c$ it is the case that for *infinitely* many values of n,
$T(n) > c' \cdot n^{\log_3 4}$. [5]

HINT: First prove that for every n of the form $2 \cdot 3^k$ $(k \in \mathbb{N})$,
$T(n) \geq c \cdot n^{\log_3 4} - 3 \cdot n - 1$.

(d) **Extra Credit:** Prove that for *all* $n \geq 1$, $T(n) \leq c \cdot n^{\log_3 4}$ where c is the constant from the previous part. (Hence, this constant c is the best possible.) [5]

4. For this question, all functions will be assumed to be functions mapping $\mathbb{N}$ to $\mathbb{N}$. We also assume the following notation: if h_1 and h_2 are functions, then
$h_1 + h_2$ is the function such that
$(h_1 + h_2)(x) = h_1(x) + h_2(x)$, and $h_1 \cdot h_2$ is the function such that
$(h_1 \cdot h_2)(x) = h_1(x) \cdot h_2(x)$.

For each of the following parts, either prove the Conjecture is true, or prove the Conjecture is false.

(a) *Conjecture:* For all functions f_1, f_2, g_1, g_2, if $f_1 \in O(g_1)$ and $f_2 \in O(g_2)$, then $f_1 + f_2 \in O(g_1 + g_2)$. [4]

(b) *Conjecture:* For all functions f_1, f_2, g_1, g_2, if $f_1 \in O(g_1)$ and $f_2 \in O(g_2)$, then $f_1 + f_2 \in O(g_1 \cdot g_2)$. [4]

(c) *Conjecture* For all functions f_1, f_2, g_1, g_2, if $f_1 + f_2 \in O(g_1 + g_2)$, then either $f_1 \in O(g_1)$ or $f_2 \in O(g_2)$. [4]