(c) Let $S_b(n)$ denote the sum of the base-b digits in n. Prove that the exact power of 2 that divides $n!$ is equal to $n - S_2(n)$. Find and prove a similar formula for the exact power of an arbitrary prime p that divides $n!$.

5. Find $d = g.c.d.(360, 294)$ in two ways: (a) by finding the prime factorization of each number, and from that finding the prime factorization of d; and (b) by means of the Euclidean algorithm.
6. For each of the following pairs of integers, find their greatest common divisor using the Euclidean algorithm, and express it as an integer linear combination of the two numbers:
(a) 26, 19; (b) 187, 34; (c) 841, 160; (d) 2613, 2171.
7. One can often speed up the Euclidean algorithm slightly by allowing divisions with negative remainders, i.e., $r_j = q_{j+2}r_{j+1} - r_{j+2}$ as well as $r_j = q_{j+2}r_{j+1} + r_{j+2}$, whichever gives the smallest r_{j+2}. In this way we always have $r_{j+2} \leq \frac{1}{2}r_{j+1}$. Do the four examples in Exercise 6 using this method.
8. (a) Prove that the following algorithm finds $d = g.c.d.(a, b)$ in finitely many steps. First note that $g.c.d.(a, b) = g.c.d.(|a|, |b|)$, so that without loss of generality we may suppose that a and b are positive. If a and b are both even, set $d = 2d'$ with $d' = g.c.d.(a/2, b/2)$. If one of the two is odd and the other (say b) is even, then set $d = d'$ with $d' = g.c.d.(a, b/2)$. If both are odd and they are unequal, say $a > b$, then set $d = d'$ with $d' = g.c.d.(a - b, b)$. Finally, if $a = b$, then set $d = a$. Repeat this process until you arrive at the last case (when the two integers are equal).
(b) Use the algorithm in part (a) to find $g.c.d.(2613, 2171)$ working in binary, i.e., find

$$g.c.d.((101000110101)_2, (100001111011)_2)$$

(c) Prove that the algorithm in part (a) takes only $O(log^2 a)$ bit operations (where $a > b$).
(d) Why is this algorithm in the form presented above not necessarily preferable to the Euclidean algorithm?
9. Suppose that a is much greater than b. Find a big-O time estimate for $g.c.d.(a, b)$ that is better than $O(log^3 a)$.
10. The purpose of this problem is to find a "best possible" estimate for the number of divisions required in the Euclidean algorithm. The *Fibonacci numbers* can be defined by the rule $f_1 = 1$, $f_2 = 1$, $f_{n+1} = f_n + f_{n-1}$ for $n \geq 2$, or, equivalently, by means of the matrix equation $\begin{pmatrix} f_{n+1} & f_n \\ f_n & f_{n-1} \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}^n$.
(a) Suppose that $a > b > 0$, and it takes k divisions to find $g.c.d.(a, b)$ by the Euclidean algorithm (the standard version given in the text, with nonnegative remainders). Show that $a \geq f_{k+2}$.