Problem 1. Calculate the g.c.d. of two positive integers, $a \geq b \geq 0$.

Solution: the Euclidean algorithm.

Pseudocode 1A.

0 Initialize: $A := a$, $B := b$
1 while $B \geq 1$
2 division: $A = Bq + R$, $0 \leq R \leq B - 1$
3 $A := B$, $B := R$
4 end(while)
5 return $A$

The correctness of the algorithm follows from the following loop invariant:

$$\text{g.c.d.}(A, B) = \text{g.c.d.}(a, b).$$

(In addition, at the end we use the fact that $\text{g.c.d.}(A, 0) = A$.)

The efficiency of the algorithm follows from the observation that after every two rounds, the value of $B$ is reduced to less than half. (Prove!) This implies that the number of rounds is $\leq 2n$ where $n$ is the number of binary digits of $b$. Therefore the total number of bit-operations is $O(n^3)$, so this is a polynomial-time algorithm. (Good job, Euclid!)

Pseudocode 1B: recursive.

0 procedure g.c.d.($a, b$) ($a \geq b \geq 0$)
1 if $b = 0$ then return $a$
2 else division: $a = bq + r$, $0 \leq r \leq b - 1$
3 return g.c.d.($b, r$)

(This code does not require a separate analysis except to clarify that it encodes the same algorithm. Clarify!) (OVER)
Problem 2. Calculate \( a^b \mod m \) where \( a, b, m \) are integers, \( a, m \geq 1, b \geq 0 \).

Solution: the method of repeated squaring.

Pseudocode 2A.

0 Initialize: \( X := 1, B := b, A = (a \mod m) \)
1 \( \text{while } B \geq 1 \) do
2 \( \text{if } B \text{ odd then } B := B - 1, X := (AX \mod m) \)
3 \( \text{else } B := B/2, A := (A^2 \mod m) \)
4 \( \text{end(while)} \)
5 return \( X \)

The correctness of the algorithm follows from the following loop invariant:

\[ X A^B \equiv a^b \mod m. \]

The efficiency of the algorithm follows from the observation that after every two rounds, the value of \( B \) is reduced to less than half. (Prove!) This implies that the number of rounds is \( \leq 2n \) where \( n \) is the number of binary digits of \( b \). Moreover, we never deal with integers greater than \( m \). Therefore the total number of bit-operations is \( O(n(\log m)^2) \leq O((\log a + \log b + \log m)^3) \), so this is a polynomial-time algorithm: the length of the input is the total number of bits of \( a, b, m \), which is \( \approx \log a + \log b + \log m \).

Pseudocode 2B: recursive.

0 procedure \( f(a, b, m) = (a^b \mod m) \) \( (b \geq 0, a, m \geq 1) \)
1 \( \text{if } b = 0 \) then return 1
2 \( \text{elseif } b \text{ odd then return } a \cdot f(a, b - 1, m) \mod m \)
3 \( \text{elseif } b \text{ even then return } f((a^2 \mod m), b/2, m) \)

(This code does not require a separate analysis except to clarify that it encodes the same algorithm. Clarify!)

Note. For both problems, the explicit (nonrecursive) versions of the algorithms are preferable.