

1. Your goal is to familiarize yourself with Appendix A of the Garey & Johnson.
2. (Related to Sipser 7.12) Let

$$\text{MODEXP} = \{\langle a, b, c, p \rangle : a, b, c \text{ and } p \text{ are binary integers such that } a^b \equiv c \pmod{p}\}$$

We wish to show that  $\text{MODEXP} \in P$ .

- (a) Explain why the following reasoning is fallacious: An algorithm can compute  $a^b$  in  $b - 1$  multiplications, take the result modulo  $p$  using one integer division, and then check if the result is equal to  $c$ .

The reasoning is fallacious since  $b - 1$  multiplications takes time  $O(2^l)$ , where  $l$  is the length of  $\langle b \rangle$ . Recall that since  $b$  is written with  $l = \lg b$  bits, and hence  $b = 2^l$ . So this algorithm takes exponential time in the length of the input  $\langle a, b, c, p \rangle$ .

- (b) Give an algorithm proving  $\text{MODEXP} \in P$ . Hint: You've seen the technique before in MCS-177. You'll need to recall the following facts about mods:

$$\begin{aligned} x + y &\equiv (x \bmod p) + (y \bmod p) && \pmod{p} \\ xy &\equiv (x \bmod p)(y \bmod p) && \pmod{p} \end{aligned}$$

Use repeated squaring or the efficient recursive algorithm you learned in MC-27. One such efficient scheme program would be:

```
(define modexp
  (lambda (a b c p)
    (define modexp
      (lambda (a b)
        (modulo
          (cond ((= b 0) 1)
                ((even? b) (square (modexp a (/ b 2))))
                ((odd? b) (* a (modexp a (- b 1)))))
          p)))
    (= (modexp a b) (modulo c p))))
```

At least every other call is with an even  $b$  (since any recursive call with odd  $b$  is followed by one with even  $b$ .) Hence, each such call shortens  $b$  by one bit, and  $O(n)$  recursive calls are required. Each call does a constant number of  $O(n)$ -bit integer arithmetic operations, and so the procedure takes polynomial time.