Exercise 1.17. The exponentiation algorithms in this section are based on performing exponentiation by means of repeated multiplication. In a similar way, one can perform integer multiplication by means of repeated addition. The following multiplication procedure (in which it is assumed that our language can only add, not multiply) is analogous to the `expt` procedure:

```
(define (* a b)
  (if (= b 0)
      0
      (+ a (* a (- b 1)))))
```

This algorithm takes a number of steps that is linear in `b`. Now suppose we include, together with addition, operations `double`, which doubles an integer, and `halve`, which divides an (even) integer by 2. Using these, design a multiplication procedure analogous to `fast-expt` that uses a logarithmic number of steps.

Answer.

```
(define (fast-mult a b)
  (cond ((= b 0) 0)
        ((even? b) (fast-mult (double a) (halve b)))
        (else (+ a (fast-mult a (- b 1))))))
```