

Write a procedure that computes the product of two non-negative integers, using only the operation of addition, and simple tests.

Hint: $x*y$ for positive x, y can be defined as

$$\begin{array}{ll} 0 & \text{if } x=0, \\ y + (x-1)*y & \text{otherwise} \end{array}$$

ANSWER:

```
(define (mult x y)
  (if (= x 0)
      0
      (+ y (mult (- x 1) y)))))
```

Exponentiation: Define a procedure that given as input integers b and n , calculates b^n ($b^n = b$ raised to the power n), using simple tests and the multiply operator ‘*’

Hint: b^n can be defined as:

$$\begin{array}{ll} 1 & \text{if } n=0, \\ b * b^{(n-1)} & \text{otherwise} \end{array}$$

ANSWER:

```
(define (expt b n)
  (if (= n 1)
      b
      (* b (expt b (- n 1)))))
```

How many multiply operations are required to calculate 5^{1000} ?

Exponentiation 2: Can you think of another procedure definition for exponentiation which takes fewer multiply operations?

Hint: We can write

$$\begin{array}{ll} b^n & = 1 & \text{if } (n = 0) \\ & = (b^{(n/2)})^2 & \text{if } (n \text{ is even}) \\ & = b * (b^{(n-1)}) & \text{if } (n \text{ is odd}) \end{array}$$

Assume we have procedures

(even? x) which returns true if x is even;

(square x) which calculates $x*x = x^2$

ANSWER:

```
(define (expt b n)
  (cond
    ((= n 0) 1)
    ((even? n) (square (expt b (/ n 2))))
    (else (* b (expt b (- n 1)))))
```

```

=====
Write a procedure that computes the Fibonacci numbers:
the n'th number fib(n) is defined as

fib(n) = 0                if (n=0)
        = 1                if (n=1)
        = fib(n-1) + fib(n-2)  otherwise

```

ANSWER: (but it's very inefficient!!)

```

(define (fib n)
  (cond
    ((= n 0) 0)
    ((= n 1) 1)
    (else (+ (fib (- n 1)) (fib (- n 2))))))

```

```

=====
Recall the recursive procedure for calculating the sum of the
integers between x and y inclusive:

```

```

(define sum (lambda (x y)
  (if (> x y)
      0
      (+ x (sum (+ x 1) y))))))

```

Now write a procedure that computes the sum of the integers between x and y inclusive, but where the process generated by the procedure should be *iterative*.

ANSWER:

```

(define (sum-iter i sum y)
  (if (> i y)
      sum
      (sum-iter (+ i 1) (+ sum i) y)))

(define (sum x y) (sum-iter x 0 y))

```

```

=====
Write a procedure that computes the product of two non-negative integers,
using only the operation of addition, and simple tests. The process
generated by the procedure should be iterative.

```

ANSWER:

```
(define (mult-iter i sum x y)
  (if (= i x)
      sum
      (mult-iter (+ i 1) (+ sum y) x y)))

(define (mult x y) (mult-iter 0 0 x y))
```

=====

Write a procedure that computes the Fibonacci numbers,
but whose process is iterative

ANSWER: (much more efficient...)

```
(define (fib-iter i a b n)
  (if (= i n)
      b
      (fib-iter (+ i 1) (+ a b) a n)))

(define (fib n) (fib-iter 0 1 0 n))
```