6.001 Tutorial 2

Gerald Dalley
14/15 February 2005
Web location:
`http://people.csail.mit.edu/~dalleyg/6.001/tutorial2.html`

Lambdas Review

➔ What do lambdas return?

➔ Is the body in a lambda expression evaluated?

Substitution Model

➔ What is the substitution model?

➔ According to the substitution model, what happens when a combination is evaluated?
1)

2)

3)

Recursion and Iteration

➔ What is a recursive *procedure*?

➔ What are the 3 parts to a recursive procedure?
1)
2)
3)

➔ What are the design steps for a recursive procedure?

➔ What is a deferred operation?

➔ What is an iterative process?

➔ Give an example of a procedure that evokes an iterative process.

➔ What is a recursive *process*?

Problems

General Problems

➔ Implement using `inc` and `dec`. Recursive or iterative?
(define (add x y)

Time order of growth:
Space order of growth:

➔ Write a recursive `slow-mul`, which multiplies two numbers by repeated addition. It need not handle negative numbers, though a simple extension would allow them.
(define (slow-mul x y)

Time order of growth:
Space order of growth:

➔ Rewrite `slow-mul` as an iterative procedure.
(define (slow-mul-iter x y)

Time order of growth:
Space order of growth:

➔ What is the order of growth of `prime?` if `sqrt` takes *Θ(1)* time?

```
(define mod
  (lambda (x y)
    (if (< x y)
        x
        (mod (- x y) y))))

(define divisible?
  (lambda (x y)
    (= (mod x y) 0)))

(define prime?
  (lambda (p)
    (define helper
      (lambda (n)
        (cond ((> n (sqrt p))   #t)
              ((divisible? p n) #f)
              (else
(helper (+ n 1))))))
    (helper 2)))
```

Biggie Size

Suppose we're designing an point-of-sale and order-tracking system for Wendy's†. Luckily the Über-Qwuick drive through supports only 4 options: Classic Single Combo (hamburger with one patty), Classic Double With Cheese Combo (2 patties), and Classic Triple with Cheese Combo (3 patties), Avant-Garde Quadruple with Guacamole Combo (4 patties). We shall encode these combos as 1, 2, 3, and 4 respectively. Each meal can be *biggie-sized* to acquire a larger box of fries and drink. A *biggie-sized* combo is represented by 5, 6, 7, and 8 respectively.

†6.001 and MIT do not endorse and are not affiliated with Wendy's in any way. They merely capitalize on the pleasant way "biggie-size" rolls off the tongue.

➔ Write a procedure named `biggie-size` which when given a regular combo returns a `biggie-sized` version.

➔ Write a procedure named `unbiggie-size` which when given a `biggie-sized` combo returns a non-`biggie-sized` version.

➔ Write a procedure named `biggie-size?` which when given a combo, returns true if the combo has been `biggie-sized` and false otherwise.

➔ Write a procedure named `combo-price` which takes a combo and returns the price of the combo. Each patty costs $1.17, and a `biggie-sized` version costs $.50 extra overall.

➔ An order is a collection of combos. We'll encode an order as each digit representing a combo. For example, the order 237 represents a Double, Triple, and `biggie-sized` Triple. Write a procedure named `empty-order` which takes no arguments and returns an empty order.

➔ Write a procedure named `add-to-order` which takes an order and a combo and returns a new order which contains the contents of the old order and the new combo. For example, `(add-to-order 1 2)` → `12`.

➔ Write a procedure named `order-size` which takes an order and returns the number of combos in the order. For example, `(order-size 237)` → `3`. You may find `quotient` (integer division) useful.

➔ Write a procedure named `order-cost` which takes an order and returns the total cost of all the combos. In addition to `quotient`, you may find `remainder` (computes remainder of division) useful.