

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
 Department of Electrical Engineering and Computer Science
 6.001—Structure and Interpretation of Computer Programs
 Spring 2004

Recitation 21
Derived Special Forms

Scheme

1. Syntax

- (a) (`quasiquote expr`) - Like `quote`, but can selectively evaluate pieces. Can be abbreviated with `'`. `Quasiquote` acts just like `quote`, except where the following two operators appear in the body of the quotation:
- i. (`unquote x`) - give value of `x`. Can be abbreviated with `,`, as in `,x`.
 - ii. (`unquote-splicing x`) - give value of `x`, assume it's a list, and splice the element into the outer list. Can be abbreviated `,@`, as in `,@x`.

For example, if `foo` is bound to `#t` and `bar` is bound to `(yay rah)`:

```
'(foo bar baz)           ; (foo bar baz)
'(foo bar baz)           ; (foo bar baz)
'(',foo bar baz)         ; (#t bar baz)
'(foo ,bar baz)          ; (foo (yay rah) baz)
'(foo ,@bar baz)         ; (foo yay rah baz)
'(foo bar ,baz)          ; error: unbound variable baz
'(',(not foo) bar baz)   ; (#f bar baz)
```

As demonstrated by the last example, the unquoted expressions aren't limited to just names.

Problems

1. If `x` is bound to `3`, `y` is bound to `(5 6)`, and `z` is bound to `(7 8 9)`, use `quasiquote` to build the value `(a 1 2 3 b 4 5 6 (7 8 9) c)`.
2. if `name` and `value` are bound, use `quasiquote` to build a `define` expression that would bind the name to the value.
3. if `params` and `body` are bound, use `quasiquote` to build a `lambda` expression with the given parameters and body.

Derived Special Forms

4. Let

```
(define (let->comb exp)
```

5. Case

```
(define (case->cond exp)
```